TREBALL DE FI DE CARRERA

TÍTOL DEL TFC : Programa per al control de Matlab mitjançant la parla

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Sistemes de Telecomunicació

AUTOR: Luis Vázquez Alejandre

DIRECTOR: Gabriel Montoro López

DATA: 22 de setembre de 2015

Títol : Programa per al control de Matlab mitjançant la parla

Autor: Luis Vázquez Alejandre

Director: Gabriel Montoro López

Data: 22 de setembre de 2015

Resum

Els avenços de l'enginyeria en el processat del senyal han desenvolupat tècniques que permeten l'anàlisi dels senyals de veu. Aquests anàlisis contribueixen al desenvolupament de sistemes reconeixedors de la parla. La indústria cada cop demanda aplicacions basades en reconeixement de la parla per tenir a l'abast una interfície diferent en la comunicació home-màquina. Una tècnica utilitzada en el reconeixement de la parla és l'anàlisi cepstrum. Primer es fa ús de tècniques de processat com acondicionament del senyal, filtrat de preèmfasi i enfinestrat, per segmentar en trames on el senyal es considera estacionari per aplicar amb posterioritat un anàlisi cepstrum de cada trama. Mitjançant un tipus de transformació homomòrfica, el cepstrum permet separar components de l'envolvent espectral i obtenir unes característiques pròpies de la paraula. Aquestes caracterítiques es poden concentrar en els *coeficients Cepstrum*. Aquests coeficients considerats estàtics s'acostumen a acompanyar amb *coeficients Dinàmics* aportant informació dinàmica per complementar el conjunt de les *característiques* de la paraula. Seguidament la tècnica Dynamic Time Warping (DTW) permet una comparació entre paraules de diferent llargària. Aquesta comparació permet escollir d'entre un conjunt de paraules, la paraula més semblant a una paraula pronunciada.
Els avenços de l'enginyeria en el camp de programari de càlcul com Matlab faciliten la tasca del desenvolupament de programes i complements basats en el reconeixement de la parla. Aprofitant aquestes eines s'ha desenvolupat un programa que té com a finalitat el comandament de Matlab mitjançant el reconeixement de la parla, per oferir una alternativa que complementi la manera de relacionar-se amb el programari. El complement tracta de executar sis tasques diferents a reconèixer per mitjà de la parla. Una de les tasques inclou diferents reconeixements posant a prova la consistència del reconeixement basat en cepstrum. Es valoren els resultats tot atenent a les limitacions del sistema per cepstrum i s'obtenen les conclusions referents a l'aplicació del cepstrum en sistemes de reconeixement per petit número de paraules, i les aportacions que pot fer el sistema de reconeixement al medi ambient.

Paraules Clau: Sistema de reconeixement de la parla, Cepstrum, coeficients Cepstrum, coeficients Delta-Cepstrum, preèmfasi, enfinestrat, Dynamic Time Warping (DTW).

Title : Softwer to control Matlab by speech recognition

Author: Luis Vázquez Alejandre

Director: Gabriel Montoro López

Date: September 22, 2015

Overview

The engineering advances in signal processing techniques have been developed which enable the analysis of voice signals. These analyzes contribute to the development of systems of speech recognizers. The industry increasingly demand applications based on speech recognition for having a different interface available in man-machine communication. A technique used in speech recognition is cepstrum analysis. First is using processing techniques such as signal conditioning, preemphasis filtering and windowing for segmented into frames where the signal is considered stationary by applying cepstrum analysis after each plot. By type of homomorphic transformation the cepstrum allows separate components of the envelope and get a spectral characteristics of the word. These caracterítiques can focus on cepstrum coefficients. These coefficients are usually considered static coefficient Dynamic accompanied providing dynamic information to complement all of the features of the word. Then the technique Dynamic Time Warping (DTW) allows comparison between different words long. This comparison can choose from a set of words, the word most similar to a word pronounced.
Advances in the field of engineering calculation software as Matlab facilitate the task of developing programs based on and complements speech recognition. Taking advantage of these tools has developed a program that aims to control the Matlab using speech recognition to offer an alternative that complements the way to interact with the software. The supplement is running six different tasks by recognizing speech. One of the tasks included testing different awards consistency recognition based on cepstrum. All results are stated in response to the limitations of the system cepstrum obtained and conclusions regarding the application of cepstrum recognition systems in small number of words, and the contribution to the environment of the speech recognition system.

Al meu pare.

ÍNDEX

ÍNDEX DE FIGURES

ÍNDEX DE TAULES

INTRODUCCIÓ

Els éssers vius, des del moment que neixen, comencen a percebre variacions del seu entorn a través dels sentits.

Comença a rebre estímuls provocant una necessitat per tornar aquests estímuls. Neix així la necessitat de comunicar.

Una d'aquestes maneres de comunicació que més ha desenvolupat l'individu és la parla. La intelligència de l'home ha estat capaç de crear un conjunt de sons de manera ordenada i codificada que sap emetre i sap interpretar. Donada la complexitat, fins que una persona parla, ha d'haver passat un temps d'aprenentatge. Aquest aprenentatge és en dues direccions la parla i la escolta. En la parla ha d'aprendre a estructurar les idees al cervell i a utilitzar correctament l'aparell fonador i en l'escolta ha daprendre a distingir els diferents sons i descodificar-los al cervell. Per tant, per parlar cal aprendre i per aprendre moltes coses cal la parla.

En el grau de desenvolupament de les persones la parla té un paper primordial.

D'altra banda l'enginy de l'ésser humà l'ha portat a crear eines per relacionar-se millor amb l'entorn i augmentar la seva productivitat. En la seva evolució juga un paper important la facilitat de fer-les anar. El seu perfeccionament fa que siguin més adequades a les nostres capacitats.

L'ésser humà ha estat capaç de desenvolupar llenguatges per comunicar-se amb les eines. Aquests llenguatges també han evolucionat en la línia d'apropar-se més al llenguatge de les persones.

Actualment els humans disposem d'eines que permeten millorar i generar altres eines. Aquest és el cas del processat digital i els programaris de càlcul i programació com ara Matlab.

El propòsit d'aquest treball és desenvolupar una manera diferent de relacionar-se amb una eina com Matlab, fent ús dels sistemes i recursos que es disposen per dissenyar un complement que per mitjà del reconeixement de la parla accioni comandaments del programari augmentant així les possibilitats en la comunicació persona - màquina.

Per la consecució d'aquesta fita, s'ha creat un programa complement de l'eina Matlab consistent en un sistema de reconeixement de la parla basat en *cepstrum*. El cepstrum real es defineix com *l'Antitransformada del logaritme del mòdul de la transformada de Fourier.* Això és:

$$c(n) = F^{-1}\{\ln|F\{y(n)\}|\}$$

Aquesta operació es considera un tipus de *transformació homomòrfica* que permet separar característiques pròpies dels *formants*. Aquestes característiques són formades pels *coeficients cepstrals*.

Els formants es consideren unes zones de freqüència corresponents a harmònics que registren màxims relatius d'intensitat originats per la corba de ressonància que correspon

al gest articulatori d'un fonema. Aquest gest articulatori és produït pel *tracte vocal*, que correspon a la part de l'aparell fonador format per cavitats de ressonància, la cavitat oral, nasal, faringe i laringe i és encarregat de filtrar els sons que provenen de la laringe per produir els diferents sons de la parla. El conjunt de formants constitueixen la funció de transferència del tracte vocal.

Per a l'obtenció dels coeficients cepstrals és necessari un processat previ del senyal. La paraula enregistrada, després de ser normalitzada i amb silencis de principi i final suprimits es sotmet a un *filtrat de preèmfasi* per compensar la pèrdua de 6 dB que experimenta des de que surt dels llavis fins que arriba al micròfon. Tot seguit és segmentada en *trames* fent servir finestres *Hamming* solapades:

$$W_{Hamming}[n] = \begin{cases} 0,54 - 0,46\cos\frac{2\pi n}{M-1}, & per\ 0 \leq n \leq M-1 \\ 0, & altrament \end{cases}$$

A continuació a cada trama se li aplica cepstrum per quedar-se amb els 10 primers *coeficients cepstrum*. Els coeficients cepstrum aporten informació estàtica i s'acompanyaran amb altres coeficients que aportaran informació dinàmica. Aquests coeficients dinàmics s'anomenen *Delta-cepstrum*. Els Delta-cepstrum es calculen realitzant una regressió lineal sobre un número de coeficients cepstrum corresponents a un interval de duració determinada. La fórmula de regressió és:

$$\Delta c(n,t) = \frac{\sum_{k=-K}^{K} k \cdot c(n,t+k)}{\sum_{k=-K}^{K} k^2}$$

on $c(n,t)$ és la seqüència cepstral $c(n)$ estimada a partir de la trama de senyal corresponent a l'instant t i K defineix l'interval d'estimació, des de $t-K$ fins a $t+K$.

Un cop es tenen les característiques (coeficients cepstrum i delta-cesptrum) que configuren cada una de les trames, es poden comparar paraules comparant les seves característiques tot fent servir una mida de distància. La distància *DTW* del Dynamic Time Warping permet comparar seqüències de mides diferents. Utilitzant el cost total de la *matriu de costos acumulats del DTW* s'obtindran les distàncies corresponents a la comparació d'una paraula amb les diferents paraules que conformen un vocabulari. Escollint la distància més curta s'identificarà la paraula més semblant.

El programa creat amb Matlab ofereix una interfície senzilla i pràctica on després d'entrenar el sistema amb les diferents paraules que el conformen, es pot escollir una tasca a realitzar per Matlab tot prement un botó i pronunciant a continuació la paraula de la tasca corresponent. El treball realitzat serveix per veure de manera pràctica els conceptes de la teoria alhora que proporciona una interfície alternativa de relació amb el programa. Els resultats són satisfactoris en vocabularis de poques paraules resultant poc aconsellable per vocabularis extensos. També s'ha pogut comprovar que els sistemes de reconeixement de la parla basats en cepstrum, encara no són aconsellables pel desenvolupament de tasques que impliquin molta precisió. Finalment es conclou amb un anàlisi de les seves limitacions i les possibles relacions amb el medi ambient.

0.1. Organització del treball

El present treball s'ha configurat en els següents capítols:

Capítol 1. Algoritmes útils en el reconeixement de la parla En aquest apartat s'exposa una breu introducció al sistema fonador productor de la parla i les diferents tècniques utilitzades per realitzar un reconeixement de la parla basat en Cepstrum.

Capítol 2. El plugin Maltlab: Aquest capítol és dedicat al programa complement de Matlab que s'ha realitzat. Consta d'apartats que parlen de les raons que impulsen a la creació d'aquest complement, dels criteris del disseny del programa i de la interfície gràfica. En apartats posteriors s'aprofundeix en les funcions del programa motor pròpiament, com s'-ha programat i es fa anàlisi de les diferents etapes que intervenen per veure de manera pràctica el processat del senyal i el cepstrum en relació amb la teoria.

Capítol 3. Resultats i valoració: En aquest capítol es comenten les diferents proves realitzades al sistema i els resultats obtinguts.

Capítol 4. Conclusions: Després de valorar els resultats, es comenten les conclusions a les que s'arriba amb la realització del treball.

Capítol 5. Influència del reconeixement de la parla en el medi ambient: Es realitza un petit estudi de com pot afectar o millorar el sistema en el medi ambient.

0.2. Paraules Clau:

Formants, tracte vocal, detecció principi final, preèmfasi, segmentació en trames, Hamming, cepstrum, delta-cepstrum, distància DTW.

CAPÍTOL 1. ALGORITMES ÚTILS EN EL RECONEIXEMENT DE LA PARLA

1.1. Sistema productiu de la veu

La paraula parlada és un so en forma vibració acústica de arriba a l'ésser humà mitjançant una propagació a través de l'aire i que és captada per l'oïda.

El sistema del cos humà corresponent a produir aquests sons és l'aparell fonador. L'aparell fonador està composat per òrgans corresponents a tres grans grups:

- Respiració
- Fonació
- Articulació

Els òrgans de respiració són els encarregats de produir aire. Aquest aire també s'utilitza per produir sons. Els òrgans són els pulmons, bronquis, bronquíols, tràquea i laringe. Al inspirar la caixa toràcica s'expandeix per mitjà de diafragma permetent el pas de l'aire exterior cap a l'interior del cos inflant els pulmons. Al expirar es realitza la operació inversa, la caixa toràcica es comprimeix i els pulmons es desinflen expulsant l'aire a través de la tràquea, laringe, boca i fosses nasals. El moment de l'expiració és aprofitat per produir els sons de la parla.

En l'expiració, l'aire viatja des del nas o la boca cap als pulmons passant per la faringe i la laringe.

El tracte vocal està constituït per cavitats de ressonància, la cavitat oral, nasal, faringe i laringe.

La laringe és una cavitat situada sobre la tràquea composada per quatre cartílags (figura 1.2):

- Cricoide: forma d'anell
- Tiroide: cartílag gran situat a la part anterior de la laringe.
- Aritenoides, grup de dos cartílags situats a la part posterior del cricoides.

Aquests cartílags poden moure's lleugerament gràcies a l'acció d'un sistema de músculs i nervis.

Dintre de la laringe es troben una espècie de membranes anomenades cordes vocals. Les cordes vocals estan constituïdes per un múscul que s'insereix per un extrem al tiroides i

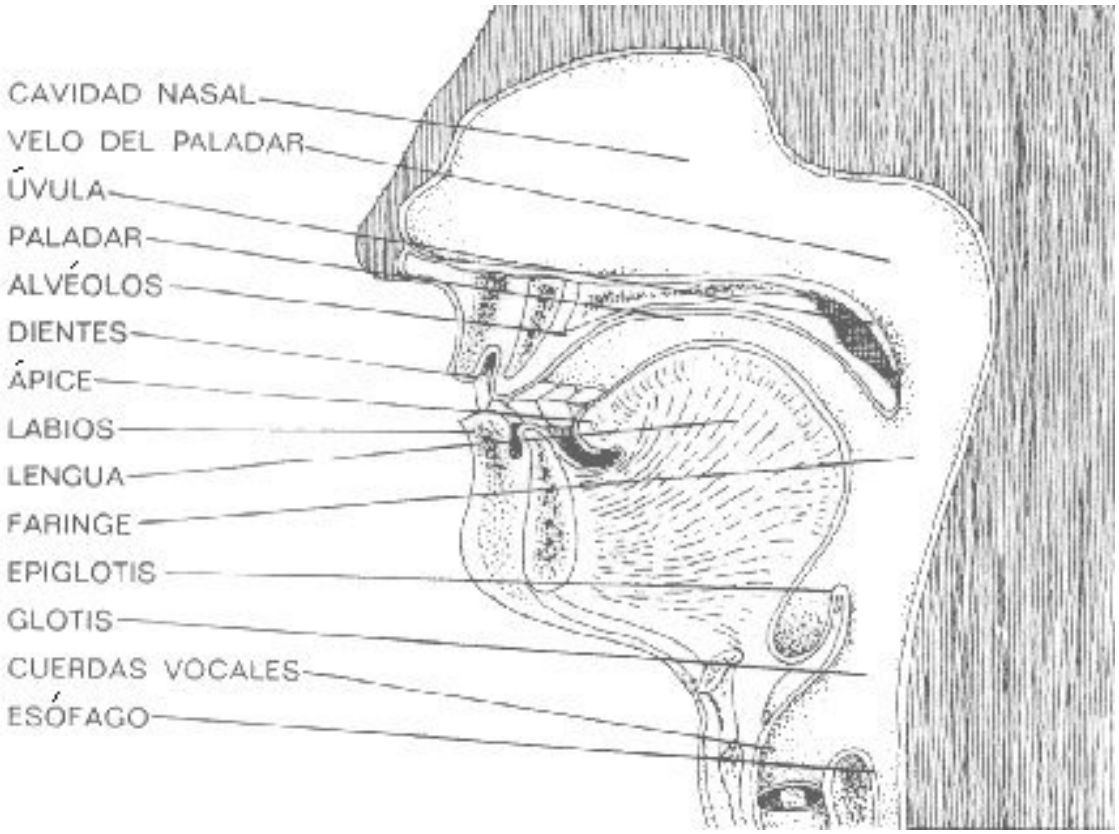

Figura 1.1: Òrgans que intervenen en la fonació. Font [20]

per l'altre extrem al ariteniodes, es tracta del múscul 'tiroaritenoide'. Un lligament molt resistent perfila les cordes vocals, aquest és el lligament vocal.

Entre les cordes vocals es forma un espai. Aquest espai és conegut com 'glotis'. La glotis pot obrir-se o tancar-se gràcies al moviment de les cordes vocals. Aquesta separació i el conseqüent moviment és produït pel moviment dels cartílags aritenoides. Aquests cartílags en separar-se o unir-se obren o tanquen les cordes vocals proporcionant d'aquesta manera més o menys espai a la glotis. Quan els aritenoides s'ajunten i es tensen les cordes vocals, la glotis es tanca totalment. El so és produït per la vibració de les cordes vocals.

A l'hora d'ingerir aliments, aquests han d'arribar a l'esòfag, passant per la faringe i per sobre de la glotis. En el moment de pas dels aliments per sobre de la glotis, la 'epiglotis' la cobreix per evitar que els aliments entrin en la tràquea.

Els sons que es produeixen a la laringe són modificats per unes cavitats que fan de ressonardors, per modificar components del so com el timbre i també produir diferents sorolls del llenguatge. Aquestes cavitats són les cavitats supraglòtiques i estan composades per les cavitats nasal, oral i faríngia.

La cavitat faríngia (faringe) és situada per sobre de la laringe i a la part posterior de la cavitat oral. El seu estretament proporciona sons consonants.

La cavitat nasal és situada per sobre de la faringe. Aquí destaca el 'vel del paladar' el

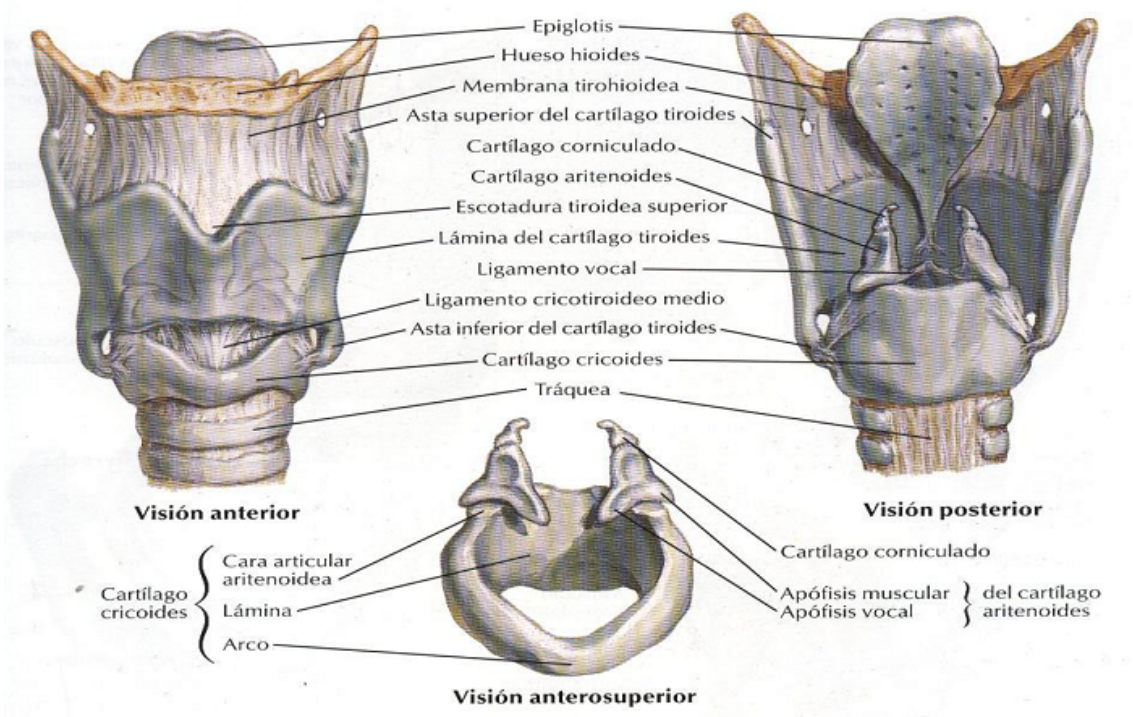

Figura 1.2: Laringe, font [19]

qual pujant o baixant pot permetre o barrar el pas de l'aire a la cavitat nasal. Comunicant la cavitat oral amb la nasal s'aconsegueixen ressonàncies com ara la que es produeix en la pronunciació en la lletra 'm' o la lletra 'o'. Tancant el pas no es produeix ressonància nasal, diferenciant d'aquesta manera els sons orals dels nasals.

La cavitat situada sota de la nasal és la cavitat oral. En aquesta cavitat és on es produeixen la majoria dels sons de la parla. La cavitat oral és utilitzada com a ressonador per modificar el timbre que prové de la laringe per produir sons com les vocals i també per produir sons utilitzats pel llenguatge corresponents a les consonants.

Dels components de la cavitat oral, destaquen:

- Llavis
- Dents
- Alvèols
- Paladar
- Vel del paladar i úvula
- Llengua

La cavitat oral es pot modificar per mitjà de gests articulatoris. Aquests són formats a voluntat tot movent parts com la llengua (moviment vertical i horitzontal) o els llavis permetent diferents configuracions o gestos de la cavitat oral que donen lloc a ressonàncies

que produeixen diferents sons. Altres sons produïts pròpiament per la cavitat vocal (corresponents a consonants) poden anar acompanyats de sons provinents de les cordes vocals o no. Aquesta distinció és la que fa possible la producció del diferents sons de consonants separades en aquests dos grans grups: sonors o sords.

Es considera doncs que el tracte vocal modifica els sons provinetns de la laringe per produir la parla. Aquesta modificació és regulada a voluntat per produir diferents sons. Així doncs, es considera el tracte vocal com un filtrat que es fa del so provinent de la laringe. Per tant tenim per una banda l'espectre glotal i per l'altra la modificació del filtre corresponent al tracte vocal. En la producció dels sons vocals el l'espectre de la laringe és modificat per la corba de ressonància corresponent al gest articulatori. Aquesta ressonància genera uns màxims relatius d'intensitat a uns determinats harmònics corresponents a unes zones de freqüència. Aquestes zones de ressonància es denominen **formants**[8].

Pel que fa a l'enteniment d'una paraula, els primers formants són els més importants, donat que segons[8] "les freqüències dels dos primers formants determinen la identitat de la major part de les vocals". La freqüència dels formants que segueixen i la freqüència fonamental són característiques més pròpies del locutor. A cada formant destaca la freqüència central i la mesura de distribució de l'energia al seu voltant, és a dir de l'ample de banda. Per tal que una paraula pugui ser intel·ligible [8] es considera que la freqüència formàntica no hauria de apartar-se més del 10% dels valors correctes.

En el reconeixement de la parla, els diferents formants juguen un paper primordial donat que el tracte vocal es comporta com un filtre que modifica el senyal provinent de la laringe per produir així el so de la paraula tal com la sentim i és el conjunt de formants els que constitueixen la funció de transferència d'aquest filtre[8].

1.2. Cepstrum

Una paraula, és un so audible per l'ésser humà, i com a tal, està constituït per un conjunt d'ones de diferents freqüències. Una manera que disposem per poder distingir les diferents freqüències que el componen és per mitjà de la representació del seu espectre. L'eina que ens permet realitzar una anàlisi espectral és la Transformada de Fourier.

La modificació del tracte vocal proporciona els diferents sons que produeixen els diferents formants. Cada configuració diferent del tracte vocal es comporta com un filtre que proporciona un so diferent. Per tant cada so diferent és produït per un filtre diferent. Podem entendre doncs, que una paraula es produeix pel pas del senyal que genera les cordes vocals per la successió de tots aquests filtres que formaran els diferents sons consecutius.

Cada un d'aquests filtres es pot modelar segons el diagrama 1.3. on $x(n)$ és el so proporcionat per les cordes vocals un cop ha passat per la faringe que entra al tracte vocal, $y(n)$ és la porció del so de la paraula i $H(z)$ és la funció de transferència del filtre. En aquest punt és important destacar que $y(n)$ és una porció molt petita de la paraula i per tant es pot considerar un senyal pràcticament estacionari. El conjunt de formants constitueixen la funció de transferència del tracte vocal[8, pàg. 25].

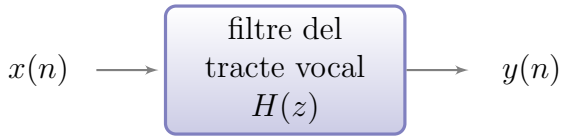

Figura 1.3: Funció de transferència del tracte vocal.

Podem tenir una porció de la paraula $y(n)$ i ens interessarà obtenir la informació relativa a la funció de transferència $H(z)$.

Tenint:

$$y(n) = h(n) * x(n) \tag{1.1}$$

Per tant, en el domini de freqüència tindrem:

$$Y(z) = H(z) \cdot X(z) \tag{1.2}$$

1.2.1. Cepstrum complex

Tenint una seqüència estable $y(n)$ amb transformada z com $Y(z)$, el cepstrum complex de $y(n)$ es defineix com la **inversa de la transformada** z **de** $\ln(Y(z))$, tractant-se d'un sistema estable es demostra[5, pàg. 270] que el cepstrum es pot definir com:

$$c(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \ln Y(\omega) e^{j\omega n} d\omega \tag{1.3}$$

Expressant $Y(\omega)$ en termes de magnitud i fase:

$$Y(\omega) = |Y(\omega)| e^{j\theta(\omega)} \tag{1.4}$$

per tant

$$\ln(Y(\omega)) = \ln|Y(\omega)| + j\theta(\omega) \tag{1.5}$$

Substituint 1.5 en 1.3, obtenim

$$c(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} [\ln|Y(\omega)| + j\theta(\omega)] e^{j\omega n} d\omega \tag{1.6}$$

Ara podem separar l'equació 1.6 en les antitransformades de Fourier corresponents a $\ln|Y(\omega)|$ i $\theta(\omega)$

$$c_m(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \ln|Y(\omega)| e^{j\omega n} d\omega \tag{1.7}$$

$$c_\theta(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \theta(\omega) e^{j\omega n} d\omega \tag{1.8}$$

1.2.2. Cepstrum real

Hem vist en les equacions1.4 i 1.5 del cepstrum complex que l'operació logaritme ha convertit la operació de multiplicació en una suma. Aquest fet ha permès separar en les components c_m i c_θ de les equacions 1.7 i 1.8 respectivament. S'anomena **cepstrum real** a l'expressió $c_m(n)$ de l'equació 1.7.

Aquesta expressió és important donat que en sistemes de reconeixement de la parla es calcula només el cepstrum real $c_m(n)$[5] utilitzat *per separar el contingut espectral*. D'altra banda cal remarcar que no és recuperable la seqüència $y(n)$ a partir del cepstrum real $c_m(n)$ donat que falta la informació de la fase de $Y(\omega)$.

El Cepstrum real es defineix com l'Antitransformada del logaritme del mòdul de la transformada de Fourier. De forma compacta l'equació 1.7 la podem escriure com:

$$c(n) = F^{-1}\{\ln|F\{y(n)\}|\} \tag{1.9}$$

Entendrem com a F la Transformada Discreta de Fourier DFT i F^{-1} l'Antitransformada Discreta de Fourier $IDFT$.

Per tant, tornant a l'equació 1.2 i aplicant logaritmes, tenim:

$$\ln|F\{y(n)\}| = \ln|F\{h(n)\}| + \ln|F\{x(n)\}| \tag{1.10}$$

Finalment, aplicant la definició de cepstrum real de l'expressió 1.9 tindrem:

$$c_y(n) = F^{-1}\{\ln|F\{y(n)\}|\} = F^{-1}\ln|F\{h(n)\}|\} + F^{-1}\{\ln|F\{x(n)\}|\} \tag{1.11}$$

Per tant el càlcul del cepstrum de la petita seqüència $y(n)$ ens permetrà **separar** les components del filtre del tracte vocal i obtenir així la informació relativa als formants. Aquest procés de convertir una convolució en una suma per separar diferents components per superposició es considera un tipus de '*transformació homomòrfica*'[1].

Observem de manera gràfica quins canvis presenta l'espectre durant aquestes etapes a la figura 1.4.

1. Al calcular el logaritme del mòdul de l'espectre d'un senyal periòdic $|F\{y(n)\}|$ es comprimeix el marge dinàmic i es redueixen les diferències en amplitud que puguin tenir els harmònics.

2. La nova forma que presenta l'espectre es pot entendre de manera semblant a una modulació en amplitud on la *moduladora* sería l'envolvent espectral i la *portadora* seria el senyal interior quasi periòdic. Aquest senyal quasi periòdic tindria un període fonamental que ens aporta informació sobre el to fonamental del locutor i l'envolvent aporta informació sobre la modulació del tracte vocal i per tant dels diferents formants.

[1] Per aquest motiu es considera l'anàlisi cepstral com un cas especial de filtrat homomòrfic.

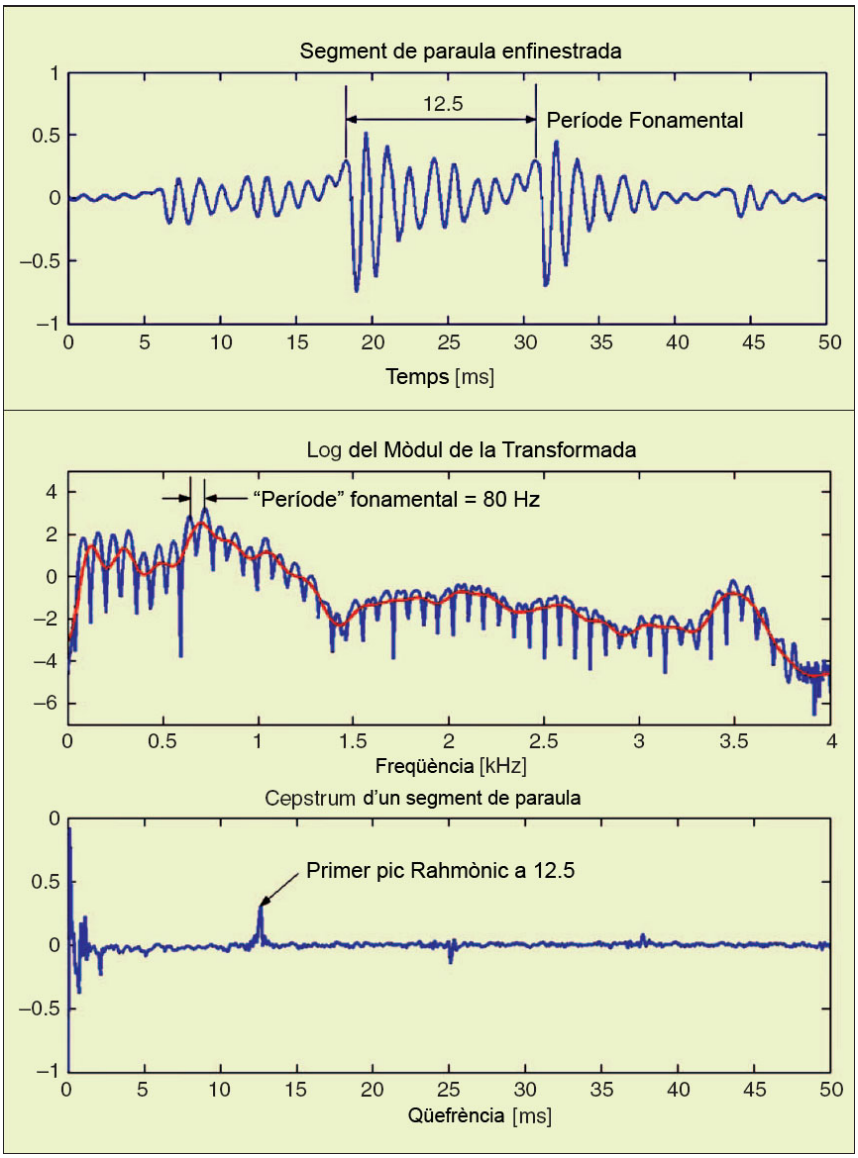

Figura 1.4: Segment de paraula enfinestrada i els canvis en l'espectre en el procés del cepstrum.[12]

3. Al fer l'antitransformada s'han separat els components de l'envolvent i del període fonamental. En aquest moment fem la proposició d'oblidar-nos que estem en el domini del temps i imaginem que estem en domini de freqüència. Com que la "moduladora" del nostre hipotètic senyal AM produïa variacions "lentes" del senyal, ocuparà temps baixos. Com que la "portadora" variava més ràpidament, ocuparà temps alts i es veurà en un gràfic en forma de petits pics semblants als tons disposats en el domini freqüencial.

La manera de mirar-se el temps des del punt de vista de freqüència, és el que va donar nom al terme '*cepstrum*' ja que prové de la inversió de les primeres lletres del terme '*spectrum*'. Amb la mateixa idea al domini del temps vist des del punt de vista de la freqüència se li anomena '*qüefrència*' i als "harmònics" que situen el tó fonamental '*rahmònics*'.

Aquests diferents components disposats en el domini del temps també anomenat com '*qüefrència*' són els **coeficients cepstrals**.

Un cop tenim la separació dels components disposats en el rang de '*qüefrències*' podem seleccionar aquells que ens interessin més. Si volem identificar persones, ens quedarem amb els components d'altes '*qüefrències*' que ens aporten el to fonamental. Si volem identificar paraules, ens quedarem amb els components de baixes '*qüefrències* que ens aporten informació sobre la modulació del tracte vocal i, per tant, sobre els formants. Al procés de seleccionar se li acostuma a anomenar '*liftat*', terme que prové d'invertir les primeres lletres de la paraula filtre en anglès ('filter').

En els sistemes de reconeixement de la parla s'acostumen a '*liftar*' dels 10 als 15 primers coeficients cepstrum. Pel nostre cas ens quedarem amb els 10 primers coeficients.

1.3. Delta Cepstrum

Hem vist que els coeficients cepstrum proporcionen informació sobre la configuració del tracte vocal. Tenir informació sobre les diferents configuracions del tracte vocal és important donat que podem tenir una idea de quins fonemes composen la paraula. No obstant, aquests coeficients ens proporcionen informació de cada segment de paraula en un moment concret, però no de la relació existent entre ells. Per aquest motiu es consideren els coeficients cepstrals com coeficients estàtics.

Hi ha un altra tipus d'informació que també resulta d'utilitat. Es tracta de valorar la relació existent entre els diferents coeficients cepstrals, de la variació que tenen en funció del temps, aportant informació sobre la variació temporal al llarg de les trames. Fer un estudi sobre la variació espectral és important donat que les zones que presenten més variació són les que aporten major informació fonètica[3]. Donat que aquesta informació fa referència a la dinàmica que tenen els coeficients cepstrals, aquests s'anomenen '*coeficients dinàmics*' o '*delta-cepstrum*'.

El concepte de variació en el temps ens porta a pensar en el concepte de derivada. Una estimació de la derivada podria consistir en restar dos valors separats en un interval de temps, però aquesta tècnica presenta problemes de soroll no desitjats[3].

Per aquest motiu s'acostuma a utilitzar la tècnica d'aplicar una regressió lineal sobre l'evolució temporal a cada coeficient cepstral en un interval de duració adequada[3]:

$$\Delta c(n,t) = \frac{\sum_{k=-K}^{K} k \cdot c(n,t+k)}{\sum_{k=-K}^{K} k^2} \tag{1.12}$$

on $c(n,t)$ és la seqüència cepstral $c(n)$ estimada a partir de la trama de senyal corresponent a l'instant t i K defineix l'interval d'estimació, des de $t-K$ fins a $t+K$.

Segons[3], la longitud d'aquest interval ha de ser prou gran per poder tenir una bona estimació de la dinàmica i suficientment breu per que modelin bé les zones de transició del senyal produïdes per la coarticulació entre fonemes. Els intervals acostumen a ser d'una duració entre el 50 ms i els 110 ms. (En el nostre cas s'ha escollit $K = 2$ proporcionant un

interval de 5 trames de 20 ms/trama amb solapament de 10 m/s proporciona un intèrval de 60 ms).

Aquests coeficients complementen als coeficients cepstrum i per tant s'acostumen a afegir al vector de característiques.

1.4. Enfinestrat

Les tècniques d'anàlisi espectral com el '*cepstrum*' permetran separar les components de l'envolvent espectral per obtenir informació sobre els formants. Aquests formants són característics d'un tipus de '*filtre*' del tracte vocal. És a dir la corba de ressonància que correspon al gest articulatori d'un fonema concret origina uns màxims relatius d'intensitat a uns harmònics corresponents a unes zones de freqüència. Per tant a partir d'un anàlisi per detectar aquestes zones de freqüència corresponents a aquests harmònics obtindrem informació de la corba de ressonància que correspon a un fonema concret.

Es tractarà doncs d'anar obtenint informació de cada un dels diferents '*gests articulatoris*' que han anat configurant cada un dels diferents fonemes de la paraula. Cada gest articulatori es pot considerar un filtre diferent o una configuració diferent del tracte vocal. Per tant el primer pas és aconseguir tenir *una porció de la paraula* corresponent a un fonema o part d'un fonema on el senyal a analitzar sigui pràcticament *estacionari*. A aquesta '*porció*' de la paraula s'aplicarà un anàlisi cepstrum on podrem obtenir informació del '*gest articulatori*' corresponent. Tot seguit es podrà obtenir informació del següent gest articulatori tot analitzant la següent porció o segment de la paraula. Per tant es tractarà de '*segmentar*' tota la paraula en porcions per anar analitzant cada una d'aquestes porcions o segments.

1.4.1. Mida de la finestra

Per tal d'obtenir cada un d'aquests segments, caldrà '*enfinestrar*' la paraula enregistrada. Aquesta finestra hauria de tenir una durada no superior al temps de duració d'un fonema. Segons estudis[11] depenent del tipus de llengua la duració dels fonemes pot variar segons la posició que ocupin en la paraula (vocals seguides de consonants sordes o sonores). Aquestes diferències però es fan més evidents en uns idiomes que en altres. En quant a la percepció, estudis[9] citen un llindar sobre els 50 als 80 mil·lisegons (ms) per sota del qual dos estímuls sonors alhora no poden distingir-se bé. Entorn als 20 ms per sons molt breus però distingibles. Pel que fa al llenguatge vocal i l'emissió de fonemes, situen la duració dels fonemes en una variació a partir d'entre 50 a 100 ms. Lehiste (1970)[10] delimita un interval entre 10 i 40 ms en el qual es pot percebre diferència entre sons (JND).[2] Mentre que M. Rossi (1971) situa aquest llindar per percebre les diferències en la duració de sons en la parla continuada entorn als 20 ms. Atenent a aquestes dades, s'acostumen a construir trames de la paraula tot enfinestrant segments d'entre 20 ms als 50 ms. Atenent a la dada de mínima percepció de distinció de sons, és usual construir trames amb finestres d'una durada d'uns 20 mil·lisegons (ms). Aquest serà el nostre cas.

[2] Llindar o interval de mínima diferència apreciable, de l'anglès '*Just Noticeable Difference*'.

1.4.2. Elecció de la finestra

En un principi, es podria pensar en anar agafant mostres de la paraula corresponents a una durada de 20 ms i configurar d'aquesta manera una trama. Aquest procediment seria com construir una trama fent servir una finestra rectangular. L'enfinestrat es configura multiplicant les mostres del senyal a enfinestrar $s(n)$ per les mostres d'una finestra $w(n)$. Ara bé, hem comentat que cal construir trames per aplicar un tractament '*cepstrum*' posterior a tota la trama i com hem vist a la secció 1.2., aquest tractament és basat en la Transformada de Fourier. Per tant al senyal se li aplicarà una multiplicació per configurar la trama i després transformada de Fourier. Atenem un moment a aquest procés:

$$s(n) \cdot w(n) \overset{DTFT}{\longleftrightarrow} \frac{1}{2\pi} S(e^{j\Omega}) \circledast W(e^{j\Omega}) \tag{1.13}$$

On el símbol $\circledast$ denota convolució periòdica[3].

En aplicar Fourier, la multiplicació esdevé convolució. D'altra banda hem d'aplicar un tractament a les mostres de la finestra per extreure unes característiques, per tant interessa que aquesta convolució afecti el menys possible a la transformada del senyal. La funció que menys afectaria al senyal a l'hora de convolucionar seria una funció '*delta*'. Si fem servir una finestra rectangular, en freqüència tindrem que la transformada del senyal es convoluciona amb la transformada del pols rectangular que és un tipus de funció '*sinc*', la qual presenta diferències a considerar respecta a una funció delta. En una finestra rectangular, els extrems presenten transició abrupta, és a dir, al enfinestrar una funció, es passa de no tenir mostres fora de la finestra a tenir-ne de cop i volta amb un 'salt' en amplitud. Aquesta discontinuïtat sobtada en l'amplitud provoca en freqüència l'aparició d'uns lòbuls secundaris de certa amplitud (forma de la funció sinc), fet que mostra la figura 1.5. Si atenem a l'amplitud d'aquests lòbuls secundaris, es pot observar que presenten poca diferència respecte al marge dinàmic en referència al lòbul principal. Al convolucionar amb la transformada del senyal, aquests lòbuls secundaris provocaran oscil·lacions espuris al senyal convolucionat, podent arribar a '*emmascarar*' components del senyal amb poca amplitud. Altra característica és la '*resolució*' en freqüència, que té a veure amb l'amplada del lòbul principal. Tal com mostra la figura 1.5 al augmentar l'amplada del pols rectangular, es redueix l'amplada del lòbul principal, augmentant així la resolució en freqüència, i les oscil·lacions que provocaran els lòbuls secundaris augmentaran la seva freqüència però no disminuiran en amplitud.

Per tant al utilitzar una finestra rectangular, tenim l'inconvenient que la seva transformada pot provocar espuris al senyal i que per millorar la resolució en freqüència caldria fer servir finestres grans, i aquest darrer punt és incompatible amb les necessitats de fer servir finestres d'uns 20 ms per enfinestrar petites porcions de la paraula per tal que el senyal sigui estacionari.

Finestres que suavitzin la transició en els extrems i presentin major diferència de marge dinàmic entre el lòbul principal i els secundaris per presentar menys espuris en freqüència, milloraria les condicions.

La figura 1.6 presenta diferents tipus de finestres definides a la taula 1.1 com ara la '*rectan-*

[3]Degut a que $S(e^{j\Omega})$ i $W(e^{j\Omega})$ són periòdiques la integració d'una convolució periòdica s'efectua sobre un període del senyal[6, p.231]

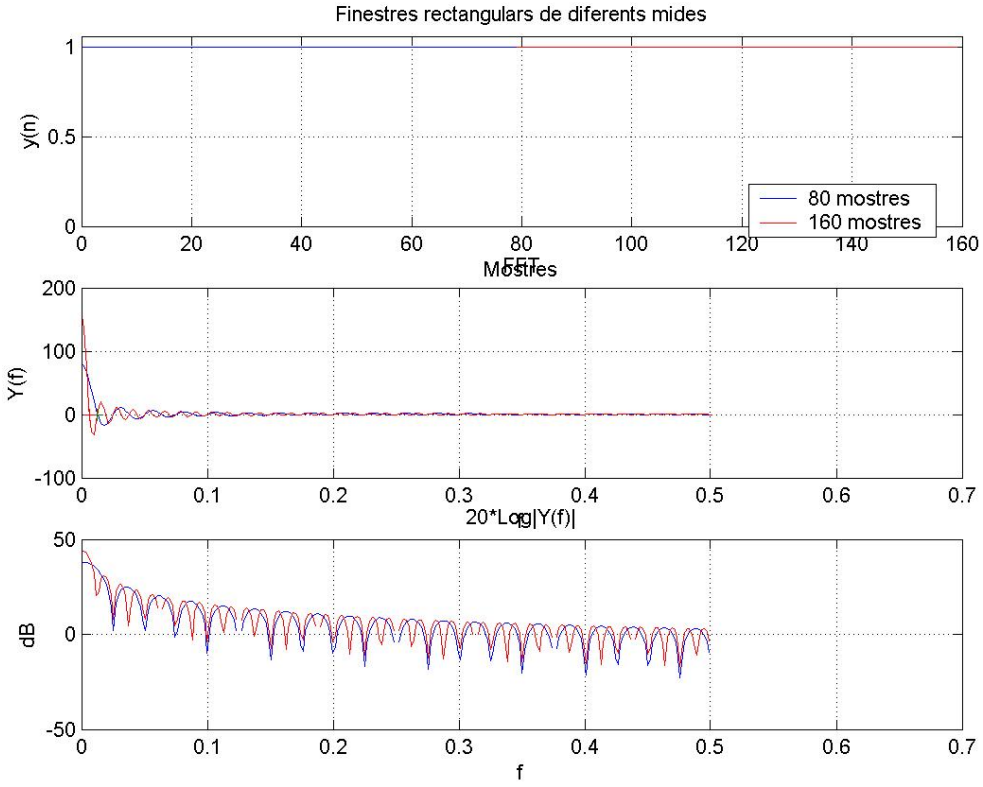

Figura 1.5: Finestra rectangular de de 80 mostres (color blau) i finestra rectangular de 160 mostres (vermell).

gular', '*Hamming*', '*Hanning*', '*Blackman*' i els respectius comportaments en freqüència.

Nom de la finestra	Seqüència en el domini temporal, $h(n), 0 \leq n \leq M-1$
Rectangular	1
Hamming	$0,54 - 0,46\cos\frac{2\pi n}{M-1}$
Hanning	$\frac{1}{2}\left(1 - \cos\frac{2\pi n}{M-1}\right)$
Blackman	$0,42 - 0,5\cos\frac{2\pi n}{M-1} + 0,08\cos\frac{4\pi n}{M-1}$

Taula 1.1: Funcions de diferents tipus de finestres d'amplada M. Per altres valors de n les funcions valen 0.

Si atenem a l'amplada del lòbul principal, veiem que en la finestra rectangular és menor, tot i que la gran presència dels lòbuls secundaris amb poca diferència de marge dinàmic respecte al principal, desvirtuen aquesta qualitat. D'altra banda l'amplada d'aquest lòbul principal és molt similar per les finestres Hamming i Hanning, tenint més amplada però presentant una major caiguda per sota dels primers lòbuls secundaris de la rectangular. Com es pot apreciar, la major amplada de lòbul principal correspon a la finestra Blackman la qual posseeix una campana més estreta en la forma de la finestra en el domini del temps (mostres). L'amplada del lòbul principal de la finestra Blackman està per sobre

dels primers lòbuls secundaris de la finestra Hamming. Tot i tenir una major atenuació en els lòbuls secundaris i per tant menys espuris, la finestra Blackman presenta una pitjor resolució en freqüència. La finestra Hanning presenta un augment d'atenuació dels lòbuls secundaris al augmentar la freqüència tot i que els primers lòbuls secundaris estan per sobre dels de la finestra Hamming. La finestra Hamming presenta major diferència de marge dinàmic entre el lòbul principal i els primers lòbuls secundaris alhora que una amplada de lòbul principal menor que la Blackman i una mica menor que la Hanning, no obstant al augmentar la freqüència els lòbuls secundaris tenen més presencia que en les finestres Hanning o Blackman. Aquest fet es degut al petit '*esglaó*' que presenta la finestra Hamming en el domini del temps (mostres) en els extrems, donat que aquests no acaben anant a zero de manera gradual com ara les finestres de Hanning o Blackman presentant una transició una mica més abrupta. Tot i així, la finestra **'*Hamming*'** representa una bona elecció al presentar més atenuació dels lòbuls secundaris que la finestra rectangular tot i tenir una major amplada del lòbul principal, però alhora aquesta amplada és menor que en les finestres Hanning o Blackman.

1.4.3. Solapament

Un cop hem seleccionat la finestra '*Hamming*', es pot procedir a la configuració d'una trama tot multiplicant les mostres de la paraula per les mostres de la finestra. Per configurar la següent trama, caldrà desplaçar la finestra i multiplicar per les mostres de la paraula o bé seleccionar les mostres de la paraula que intervenen en la següent trama i multiplicar-les per la finestra.

Ara és moment de tornar a fixar-nos en la forma que presenta la finestra Hamming en el temps (mostres). Com hem vist, s'ha escollit la finestra Hamming per que presenta més diferència en marge dinàmic del lòbul principal als lòbuls secundaris que la rectangular, i aquest fet és degut al decaïment gradual cap a zero que presenten les mostres als extrems de la finestra. Per tant al multiplicar les mostres del senyal per la finestra, la part del senyal que queda '*enfinestrada*' i que configura una trama, presenta atenuació a les mostres situades als extrems de la trama, tal com mostra la figura 1.7. Degut a aquest fet, si subdividim tota la paraula en trames i aquestes trames són contigües, hi haurà parts de la paraula corresponents als extrems de cada trama que no quedaran representades al presentar molta atenuació. Per aconseguir tenir representades les parts corresponents als extrems de cada trama, és necessari no construir les trames de forma contigua, sinó solapada. Una forma habitual és fer coincidir l'extrem esquerra (dret) d'una trama amb el punt mig de la trama anterior (posterior), amb l'objectiu de fer coincidir un mínim de la finestra amb un màxim de les finestres anteriors o posteriors, tal com mostra la figura 1.8. D'aquesta manera les parts atenuades per una trama queden compensades per les representacions en les trames anterior i posterior. Es produeix doncs un solapament de les respectives trames que configuren la paraula. En aquest cas concret si es fa coincidir un extrem de la trama amb la meitat de la trama següent o anterior hi ha un solapament de mitja trama, o bé traduït en temps, significaria que si les trames duren 20 mil·lisegons, construiríem trames cada 10 mil·lisegons.

1.5. Preèmfasi

La producció dels diferents sons que configuren la parla, es basa en el filtrat que el tracte vocal fa del so provinent de la laringe. El tracte vocal es comporta com un filtre que a mode de ressonador, farà que a unes zones de freqüència determinats harmònics presentin màxims relatius d'intensitat. Per tant el tracte vocal presenta punts de ressonància anomenats formants que modifiquen el so provinent de la laringe. A partir de la detecció d'aquestes zones de freqüència on els harmònics presenten màxims relatius s'obté informació dels formants que caracteritzen un fonema.

Hi ha fonemes que presenten formants a altes freqüències. Per realitzar un anàlisi de la paraula, aquesta cal primer enregistrar-la mitjançant un micròfon. Però quan la paraula és pronunciada, des de que surt dels llavis és atenuada uns 6 dB per octava[4]. Això significa que cada vegada que es dobla la freqüència la potència queda reduïda a la quarta part. A més a més cal recordar de l'apartat 1.4.2. que per segmentar en trames fem servir finestres i aquestes en freqüència presenten lòbuls secundaris[4] que al convolucionar amb el senyal podrien emmascarar components amb molt poca amplitud.

Donada la importància de no perdre informació relativa a altes freqüències cal una solució per compensar l'atenuació que pateixen. Aquesta és la missió del *filtre de preèmfasi*. Es tracta d'un filtre de primer ordre de tipus passa altes aplicat a la paraula abans del tractament de FFT amb una funció de transferència:

$$H(z) = \frac{Y(z)}{X(z)} = 1 - az^{-1} \tag{1.14}$$

Amb una equació en diferències com ara:

$$y(n) = x(n) - ax(n-1) \tag{1.15}$$

El coeficient de preèmfasi a acostuma a prendre valors dins l'interval entre 0,95 i 1. Augmentant l'efecte de preèmfasi a mida que a s'apropa a 1. La figura 1.9 mostra a l'esquerra la resposta en freqüència del filtre de preèmfasi amb coeficient a = 0,97 i a la dreta amb diferents valors del coeficient a.

1.6. DTW (Dynamic Time Warping)

El Dynamic Time Warping (DTW) (literalment distorsió temporal dinàmica o alineament temporal dinàmic) és una tècnica utilitzada per trobar un alineament òptim entre dues senyals dependents del temps. Mitjançant aquest alineament, aquesta tècnica es pot utilitzar per mesurar la similitud entre aquestes dues senyals que poden variar en temps o en velocitat. Donades aquestes característiques un ús habitual del DTW ha estat en el camp del reconeixement automàtic de la parla. Així doncs pensem per un moment en aquesta problemàtica. Quan un parlant pronuncia una paraula, és gairebé impossible que la torni a pronunciar amb la mateixa velocitat. Per exemple pronunciem una mateixa paraula dues vegades. És molt difícil que totes dues paraules durin exactament igual però

[4] Tot i escollir una finestra que minimitzi la presència de lòbuls secundaris, aquests hi seran presents.

en el supòsit que la paraula pronunciada per segona vegada tingués una duració en temps exactament igual a la pronunciada primera, també podria haver variacions temporals entre els diferents fonemes que la formen. És a dir, parts de la paraula segona pronunciades a diferent velocitat que les mateixes parts de la paraula primera. Per tant alhora de comparar dues paraules, és necessària la existència d'un algoritme capaç de mesurar mínimes distàncies entre senyals de diferent llargària, i el DTW ens ho permet.

1.6.1. DTW clàssic

Partim de la necessitat de comparar dues senyals discretes, dependents del temps i de llargàries diferents. Aquestes senyals les anomenarem X i Y amb llargàries $N \in \mathbb{N}$ i $M \in \mathbb{N}$ respectivament tals que:

$$X := (x_1, x_2, \ldots, x_N)$$
$$Y := (y_1, y_2, \ldots, y_M)$$

Al tenir aquestes senyals, que podem entendre com a vectors de llargàries diferents, no es pot fer una comparació component a component (x_1 amb y_1, x_2 amb $y_2 \ldots$), tampoc podem calcular una distància euclídea directament entre els dos vectors donat que $N \neq M$.

El procediment del DTW es basa en comparar cada un dels components del senyal X amb tots els components del senyal Y. En cada comparació es mesura un cost anomenat "*cost local*" o també "*distància local*". Així doncs es prenen els components x_1 i y_1, es mesura el cost c_{11}. A continuació es prenen els components x_1 i y_2 per mesurar el cost c_{12}. Així successivament es va avaluant el cost per cada parell d'elements dels senyals X i Y, donant lloc a una *matriu de costos* $C(n,m)$.

$$C = \begin{pmatrix} c_{N1} & c_{N2} & \cdots & c_{NM} \\ \vdots & \vdots & \ddots & \vdots \\ c_{21} & c_{22} & \cdots & c_{2M} \\ c_{11} & c_{12} & \cdots & c_{1M} \end{pmatrix} \tag{1.16}$$

El càlcul del cost local c_{ij} acostuma a ser una avaluació definida amb uns criteris de decisió. Per exemple podria ser un valor lògic prenent cost 0 si els components a comparar son iguals i 1 si son diferents, o bé pot tractar-se d'un càlcul de distància. Normalment s'acostuma a triar un algoritme que proporcioni un resultat del cost petit si els components a comparar son semblants i cost més elevat si son diferents.

Per tant en la matriu de costos C apareixen components amb uns valors mínims que ens estan indicant el *camí* per on els senyals comparats se semblen més. Com si d'una "sopa de lletres" es tractés, es van confeccionant *camins* que ens portin des del primer component de la matriu c_{11} fins el darrer c_{NM}. Cadascun d'aquests *camins* s'anomena *warping path* i és una seqüència $p : (p_1, \ldots, p_L)$ on cada p_l és un índex d'un component de la matriu de costos C. Aquesta seqüència p ha de satisfer tres condicions:

1. El primer element ha de ser el primer component de la matriu de costos C i el darrer element ha de ser el darrer component de C: $p_1 = (1,1)$ i $p_L = (N,M)$.

2. Cal avançar en els índex de la fila o de la columna o bé quedar-se en el mateix index de fila o columna, però mai retrocedir un índex: $n_1 \leq n_2 \leq \cdots \leq n_L$ i $m_1 \leq m_2 \leq \cdots \leq m_L$

3. S'ha d'avançar de forma contigua de tal manera que el següent element de p ha d'estar a distància d'una fila o bé d'una columna o bé d'una fila i d'una columna: $p_{l+1} - p_l \in \{(1,0),(0,1),(1,1)\}$ per $l \in [1 : L-1]$.

L'objectiu seria aconseguir un camí tot seguint els components de menor cost.

El **cost total** $c_p(X,Y)$ d'un camí '*warping path*' p entre X i Y respecte la mesura del cost local c es calcula tot sumant els diferents costos dels components que formen el camí p. Això és:

$$c_p(X,Y) = \sum_{l=1}^{L} c(x_{n_l}, y_{m_l}) \tag{1.17}$$

Per tant, un camí '*warping path*' *òptim* entre les seqüències X i Y és aquell que té el **mínim** cost total entre tots els possibles camins. Aquest cost mínim és el que es coneix com la '*distància DTW*' $\mathrm{DTW}(X,Y)$ entre X i Y.

Cal destacar que de tots els possibles camins, pot haver més d'un que tingui el cost total igual al cost mínim. Per tant en aquest cas hi hauria *més d'un camí òptim*. Pensem que per determinar un 'camí òptim', cal primer trobar tots els camins possibles entre X i Y i calcular el cost de cada camí i d'entre tots aquests quedar-se amb el o els de mínim cost. Aquest procediment tindria un cost computacional elevat que s'incrementaria a mesura que incrementem la mida de les seqüències a comparar X i Y. Per aquest motiu s'implementen algoritmes que es puguin programar de manera més eficient. Un d'ells és el que es coneix com a '*matriu de costos acumulats*'.

1.6.2. Matriu de Costos Acumulats

S'anomenarà la matriu $D(n,m)$ a la matriu de costos acumulats tal que calcularà la distància DTW entre X i Y $\mathrm{DTW}(X,Y)$. Aquesta matriu de '*costos acumulats*' tindrà a la darrera component $D(N,M)$ el mínim cost total acumulat, és a dir, la distància $\mathrm{DTW}(X,Y)$.

La matriu de costos acumulats $D(n,m)$ es calcula de la següent manera:

1. Cada component n de la primera columna és la suma dels n primers components de la primera columna de la matriu de costos C.

2. Cada component m de la primera fila és la suma del m primers components de la primera fila de la matriu de costos C.

3. Per calcular els components (n,m) de la matriu D que no siguin de primera fila ni de primera columna, es buscarà el mínim dels components de la matriu de costos

acumulats D ja calculats i que envolten a (n,m) i es sumarà al component (n,m) de la matriu de costos C.

D'aquesta manera, el darrer component de la matriu $D(N,M)$ serà el mínim cost acumulat i, per tant, la distància $DTW(X,Y)$.

Expressat de manera matemàtica:

$$\begin{array}{lcll} D(n,1) & = & \sum_{k=1}^{n} c(x_k,y_1), & per\ n \in [1:N] \\ D(1,m) & = & \sum_{k=1}^{m} c(x_1,y_k), & per\ m \in [1:M] \\ D(n,m) & = & \min\{D(n-1,m-1),D(n-1,m),D(n,m-1)\}+c(x_n,y_m), & per\ 1<n\leq N\ i \\ & & & 1<m\leq M \end{array} \quad (1.18)$$

Si només es vol calcular la distància $DTW(X,Y)$ aquesta serà el darrer component $D(N,M)$. Si es vol obtenir tot el camí òptim de cost mínim $p=(p_1,\ldots,p_L)$, es partirà del darrer component (N,M) i aquest serà el darrer component del camí p, és a dir $p_L=(M,N)$. A continuació es retrocedirà de la següent manera:

1. Si estem a la filera 1 retrocedirem a la columna immediata inferior.
2. Si estem a la columna 1 retrocedirem a la fila immediata inferior.
3. Si no estem ni a la fila 1 ni a la columna 1 seguirem el camí tot escollint el valor mínim inferior que l'envolta, bé a la fila inferior, o bé a la columna inferior, o bé a la fila i columna inferior. En cas de existir més d'una posició igual al mínim, s'escollirà al de la diagonal inferior, és a dir retrocedir fila i columna.

O bé expressat de manera matemàtica:

$$p_L = (N,M) \quad (1.19)$$

$$p_{l-1} = \begin{cases} (1,m-1), & \text{si } n=1 \\ (n-1,1), & \text{si } m=1 \\ \operatorname{argmin}\{D(n-1,m-1),D(n-1,m),D(n,m-1)\}, & \text{altrament} \end{cases} \quad (1.20)$$

Exemple de càlcul de la matriu de costos acumulats D a partir de la matriu de costos C:

$$c(X,Y)=\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad D(X,Y)=\begin{pmatrix} 3 & 2 & 2 \\ 2 & 2 & 2 \\ 1 & 1 & 2 \\ 0 & 1 & 2 \end{pmatrix}$$

Total cost acumulat i, per tant, distància DTW = 2. '*Warping Path*' òptim:

$$\begin{pmatrix} 3 & 2 & \boxed{2} \\ 2 & \boxed{2} & 2 \\ \boxed{1} & 1 & 2 \\ \boxed{0} & 1 & 2 \end{pmatrix}$$

p_l s'obté dels índex corresponents als valors $(0,1,2,2)$. Per tant: $p_l=((1,1),(2,1),(3,2),(4,3))$

1.7. Reconeixement de la parla

En els apartats anteriors hem vist les diferents tècniques que permeten obtenir unes característiques d'una paraula enregistrada que la fan diferent d'altres paraules. També hem vist una tècnica per comparar les característiques de dues paraules i obtenir una mesura de distància. Ara es tractarà de veure com es combinen tots aquests aspectes per aconseguir un sistema reconeixedor de la parla.

1.7.1. Entrenament

En primer lloc, un tret molt important és '*entrenar*' el sistema per tal que tingui unes referències amb les quals comparar. Igualment les persones necessitem una base de paraules que hem après per tal que al sentir una paraula semblant a les que coneixem, el nostre cervell la pugui identificar.

D'altra banda la fase d'entrenament inclou l'extracció de característiques de cada paraula per ser guardada a la memòria. Per tant un primer pas important és arribar a l'extracció de característiques.

Hem vist que el '*cepstrum*' permet separar l'envolvent de l'espectre el qual ens proporciona informació sobre el filtre del tracte vocal referent als formants, que són característics dels diferents fonemes. Per aquest motiu, serà important tenir una porció de paraula no més gran que un fonema i on el senyal sigui pràcticament estacionari. Per tant *prèviament* al tractament cepstrum, caldrà *segmentar la paraula en trames* d'uns 20 mil·lisegons. També hem vist que degut a les operacions implicades al cepstrum cal utilitzar finestres que suavitzin els extrems, essent una bona opció la finestra '*Hamming*' i que degut a les característiques de la finestra Hamming, caldrà '*solapar*' les finestres. També hem vist que en pronunciar la paraula el senyal perd uns 6 dB per octava, perdent senyal en altes freqüències que són necessàries perque aporten informació d'alguns formants i per tant caldrà aplicar un '*filtre de preèmfasi*' per compensar aquesta atenuació.

Per tant la seqüència de passes a realitzar per l'extracció de característiques seria el que recull el diagrama de blocs de la figura 1.10. Tot seguit enumerem les diferents etapes:

1. Enregistrament d'una paraula: mitjançant el micròfon es fa un enregistrament que sigui òptim tot complint uns mínims requisits de qualitat relació senyal soroll per resultar intel·ligible.

2. '*Front end detection*': se li acostuma a anomenar a la detecció del principi i el final de la paraula, és a dir, un processat previ per tal de tenir tots els enregistraments en les mateixes condicions. S'acostuma a normalitzar la paraula i a eliminar els silencis del principi i el final.

3. Filtre de preèmfasi: la paraula enregistrada pel micròfon ha patit una atenuació a altes freqüències des que ha sortit dels llavis. És moment de compensar aquesta pèrdua aplicant el filtre de preèmfasi abans del tractament cepstrum.

4. Segmentació en trames: mitjançant l'enfinestrat es confeccionen trames on el senyal es pugui considerar estacionari. Les trames seran d'uns 20 mil·lisegons amb solapament.

5. Cepstrum: s'extreuen les característiques relatives als formants de cada trama, tot realitzant un '*liftat*' quedant-se amb els primers 10 a 15 coeficients cepstrum per separar l'envoltant de l'espectre.

6. Delta-cepstrum: es completen les característiques estàtiques cepstrum amb les que aporten informació dinàmica. Amb aquest bloc conclou l'extracció de les característiques

7. Guardar a la base de paraules les característiques extretes de la paraula.

8. Repetir tot el procés per totes les paraules fins configurar tota la base de paraules en memòria.

La figura 1.11 recull amb detall el procés d'extracció de les característiques.

1.7.2. Reconeixement

Un cop tenim el sistema entrenat amb una base de paraules per poder comparar, el sistema pot procedir al reconeixement d'una paraula.

Per tal de reconèixer una paraula, primer cal que entri al sistema, per tant la primera fase és molt semblant a la fase d'entrenament. Per que entri la paraula al sistema, primer cal enregistrar-la, que aquest enregistrament sigui fet amb un mínim de qualitat, i a continuació es prepara per extreure les característiques. Això implica seguir les passes 1) a 6) de la fase d'entrenament: enregistrament - processat previ per detectar principi i final - aplicar el filtre de preèmfasi - segmentació en trames - extracció de coeficients cepstrum - complement dinàmic mitjançant delta-cepstrum.

Un cop es tenen les característiques de la paraula entrant, cal comparar-les amb les característiques de cada una de les paraules que el sistema té a la base. En aquest punt entra en joc el '*Dynamic Time Warping (DTW)*'. Mitjançant el càlcul de la distància DTW es podran comparar matrius de components de diferent llargària, és a dir paraules diferents i que tenen, per tant, diferent número de trames. La mesura de distància del DTW ens diu que quan dues seqüències tenen distancia menor significa que són més semblants. Per tant guardant en memòria cada una de les mesures de distància de la paraula entrant amb les diferents paraules de la base es podrà fer una cerca per trobar la '*mínima distància*' de totes les calculades. Aquesta mínima distància correspondrà a la '*paraula identificada*'.

La figura 1.12 representa el diagrama de blocs que resumeix tot el procés per dur a terme el reconeixement d'una paraula.

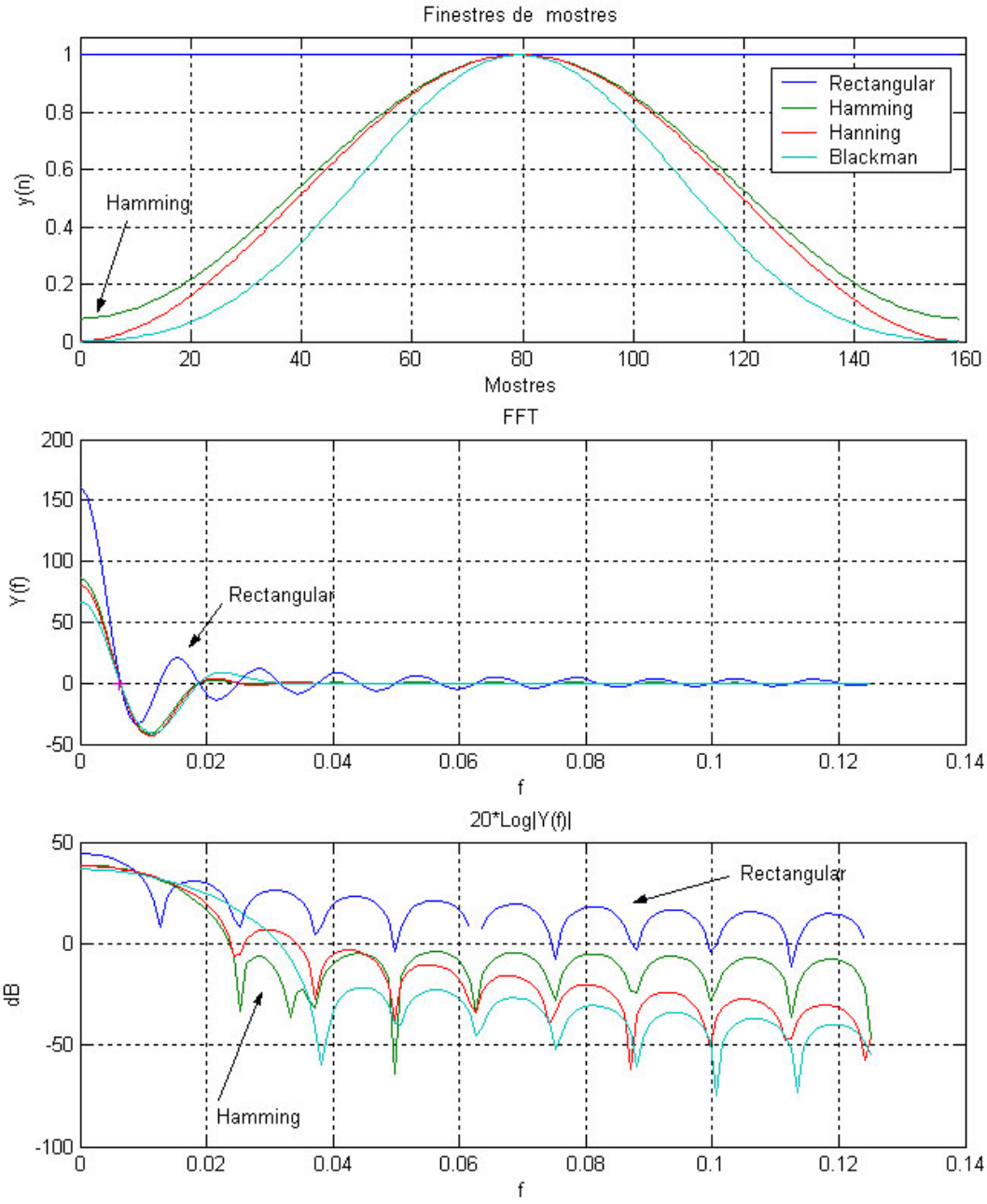

Figura 1.6: Finestres Rectangular, Hamming, Hanning i Blackman de 160 mostres.

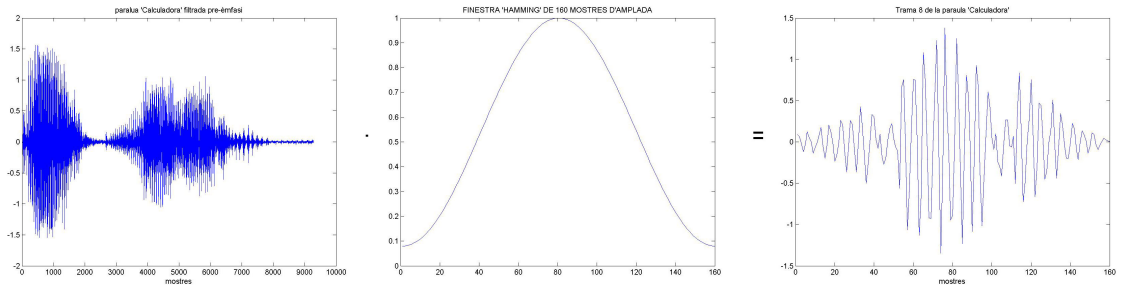

Figura 1.7: Formació d'una trama mitjançant enfinestrat.

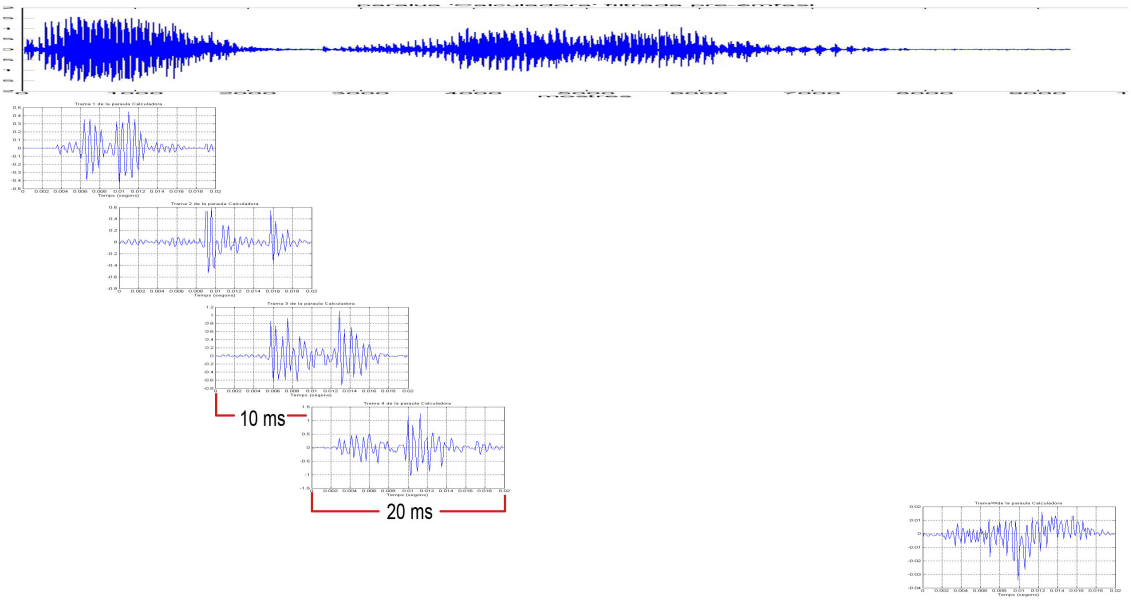

Figura 1.8: Segmentació de la paraula en trames de 20 ms cada 10 ms.

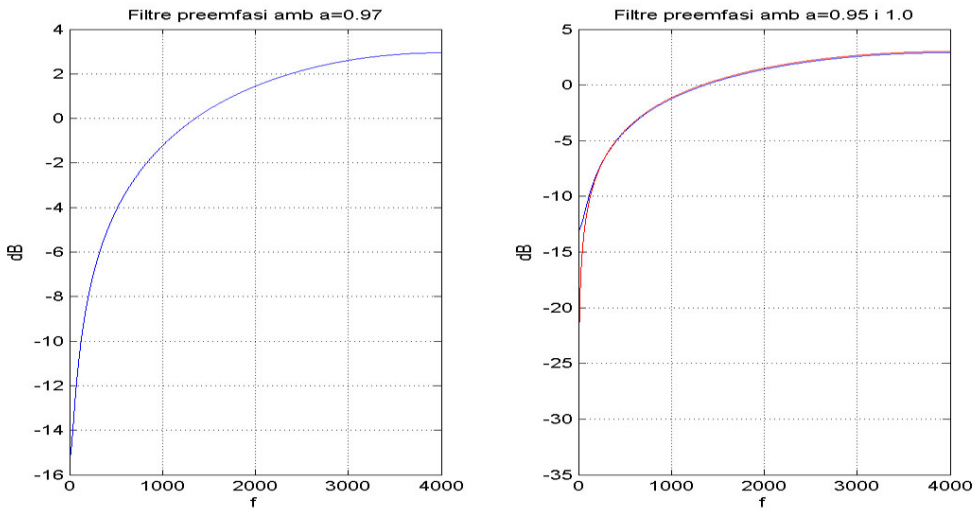

Figura 1.9: Dreta: Filtre preèmfasi a=0,97. Esquerra: Filtres preèmfasi a=0,95 (blau), a=1 (vermell).

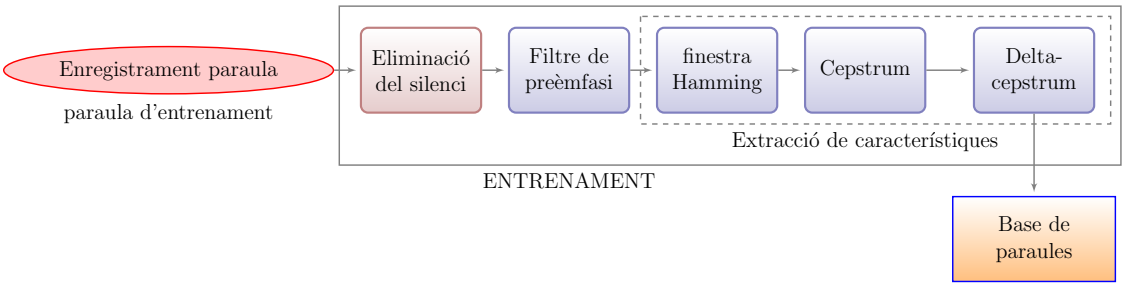

Figura 1.10: Procés d'entrenament del sistema.

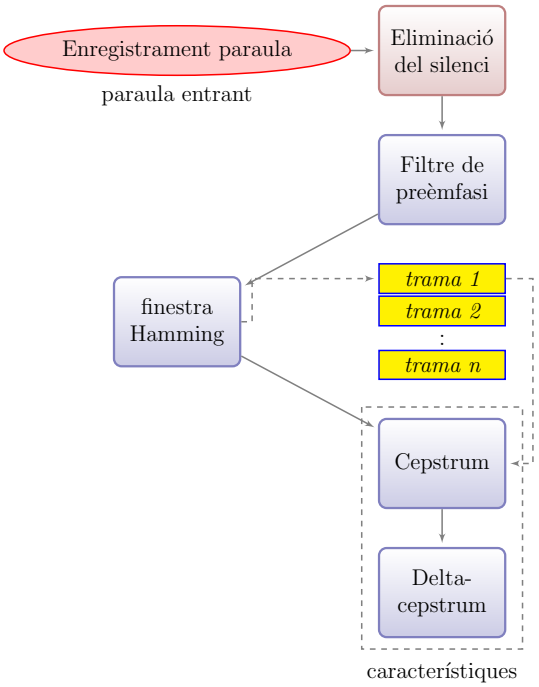

Figura 1.11: Procés d'extracció de característiques.

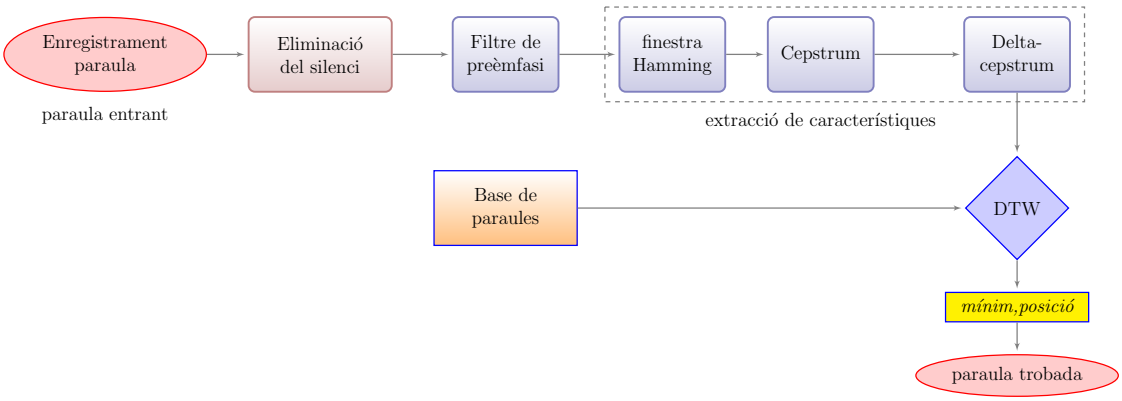

Figura 1.12: Procés de reconeixement d'una paraula.

CAPÍTOL 2. EL PLUGIN PARLAB

Matlab és un potent programari de càlcul, molt utilitzat en el àmbit científic i tecnològic. A més a més de dotar a l'usuari de potents eines de càlcul i anàlisi, poseeix la qualitat d'haver desenvolupat un propi llenguatge, mitjançant el qual es poden programar tasques complexes i funcions que posteriorment es poden executar. Aquestes funcions poden incorporar d'altres pròpies de Matlab o altres funcions que també hagi creat un usuari amb anterioritat. També disposa de modes de compatibilitat amb altres llenguatges de programació (C, Fortran), de tal manera que és possible programar funcions en altres llenguatges i que Matlab les pugui fer servir. També compta amb la possibilitat de desenvolupar interfícies gràfiques on es poden dissenyar diferents comandaments per executar les diferents rutines que s'hagin programat o interactuar amb elles. Per tant també serveix per dissenyar sistemes sensers o emuladors de sistemes que posteriorment calgui construir. Amb tot es tracta d'un programari d'un ús molt estès, molt flexible i amb un gran ventall de possibilitats.

2.1. Raó d'ésser de Parlab

Quan un usuari obre el programari Matlab (en el nostre cas Matlab 6.5), el primer que es troba és tot un seguit de finestres organitzades al voltant d'un escriptori tal com mostra la figura 2.1. En destaquen tres finestres principalment:

1. Workspace - Current Directory
2. Command History
3. Command Window

El primer sector a dalt a l'esquerra acostuma a ser compartit per dues finestres (Workspace i Current Directory) podent commutar entre elles mitjançant una pestanya associada a cadascuna d'elles i situada per sota. Com el seu nom indica '*Workspace*' ens mostra l'espai de treball, sobre el qual podem anar visualitzant la informació sobre totes i cada una de les diferents variables que anem creant. Així doncs podem saber el nom de la variable, del tipus o classe que és, la mida i els Bytes que ocupa.

Si commutem mitjançant la pestanya de sota a la finestra '*Current Directory*', es mostra el directori actual de treball del programari, és a dir, ens diu el directori des del qual Matlab intentarà carregar o a on guardarà arxius que l'usuari sol·liciti sense especificar una ruta concreta. Per tant aquesta finestra ens diu quins arxius i de quin tipus i quins subdirectoris conté. Cal dir que per tal que Matlab pugui executar una funció o obrir un arxiu o carregar unes dades, o simplement per visualitzar-les, Matlab ha de poder accedir a elles i per tant resulta d'utilitat saber en quin directori es troba actualment Matlab.

Per sota d'aquest sector o finestra, tenim una altra finestra (abaix a l'esquerra) on s'ens mostra el '*Command History*'. Com el seu nom indica, es tracta d'un historial de les

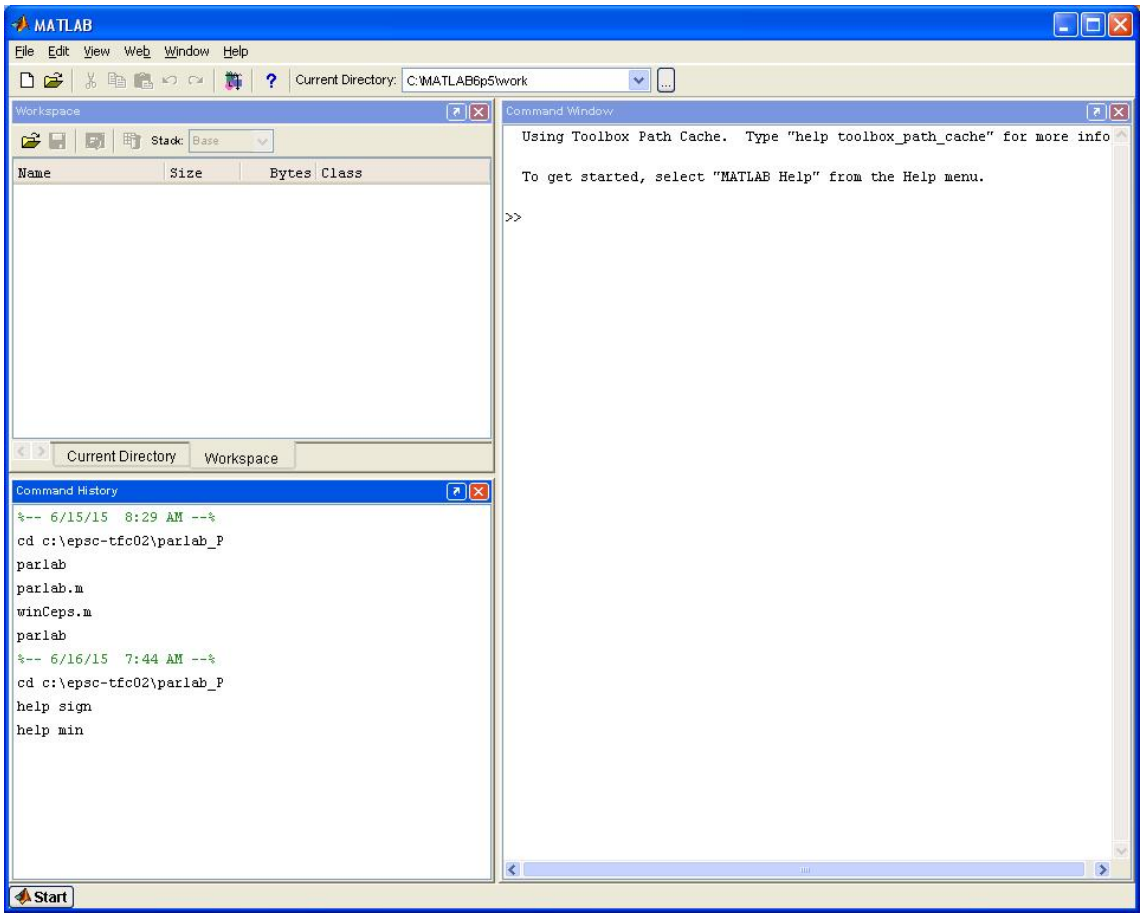

Figura 2.1: Finestra principal del programari Matlab

darreres comandes que s'han anat ordenant a Matlab així com el dia i l'hora en què s'han fet.

Arribem a la finestra principal, la més gran situada a la dreta, '*Command Window*'. Aquesta és la finestra de comandes. Aquí és on es poden definir variables, cridar funcions, activar rutines... fins i tot demanar ajuda. Tota variable que creem queda automàticament assignada i visible en el '*Workspace*'. Si, per exemple, teclejem:

```
>> a=5
```

al prémer la tecla '*return*' s'ens mostrarà:

```
>> a=5

a =

5
```

la qual cosa ens demostra que s'assigna el valor 5 a la variable `a`, i automàticament podem visualitzar en la finestra del 'Workspace' la variable creada.

D'altra banda, si volem accedir a l'ajuda d'alguna comanda concreta de Matlab, podem sol·licitar-la teclejant al '*Command Window*' la paraula '*help*' i a continuació el nom de la comanda de la qual volem obtenir ajuda. En la figura 2.2 es pot apreciar com després de

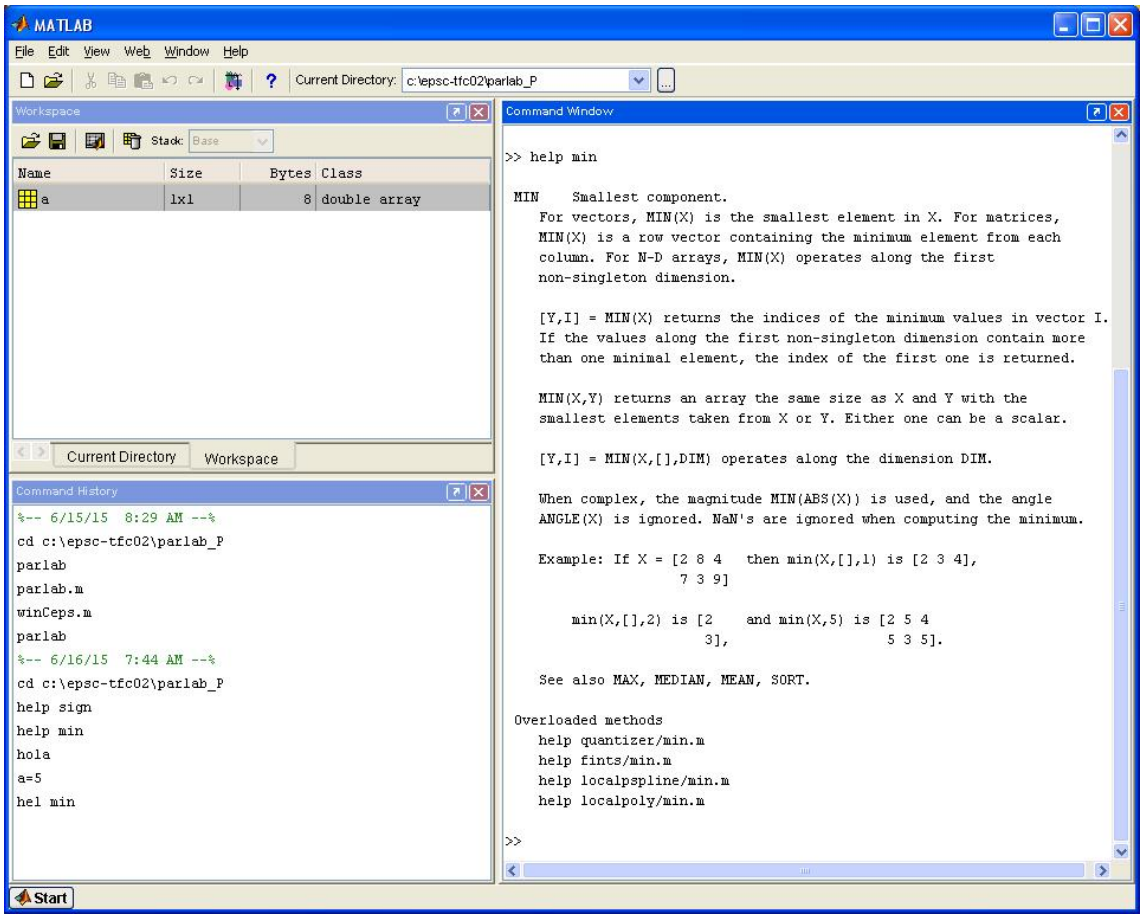

Figura 2.2: Sol·licitud d'ajuda a Matlab mitjançant el '*Command Window*'

sol·licitar ajuda sobre la funció '`MIN`' tot teclejant en la finestra '*Command Window*':

```
>> help min
```

i prement a continuació la tecla '*return*', es desplega tot el text d'ajuda sobre la funció '`MIN`' a la mateixa finestra '*Command Window*'. Per tant el '*Command Window*' serveix tant per visualitzar les comandes que nosaltres fem sobre Matlab com també per visualitzar la informació que Matlab ens proporciona.

Fins aquí, hem donat una pinzellada de la manera de com es presenta l'entorn de treball de Matlab per desenvolupar-nos amb aquest programa. Ara pot ser és moment de pensar en l'entorn en que sovint un usuari es pot trobar alhora de fer anar Matlab.

Quan obrim Matlab, moltes vegades, queda obert en una finestra petita, no a pantalla comperta, al nesessitar espai de pantalla per altres aplicacions que es tenen obertes al mateix temps, com ara un document de text, un editor de textos o processador de text per anar fent anotacions, altres finestres del propi Matlab... Significa això que en ocasions la visió de les tres finestres que ocupen l'entorn de treball, són més petites. A més a més

cal afegir que moltes de les tasques a realitzar amb Matlab es poden fer amb un ordinador portàtil, amb la qual cosa la pantalla pot resultar encara més petita. Per aquests motius de vegades l'usuari tanca les finestres de '*Workspace*' de '*Current Directory*' o de '*Command History*', per activar-les amb posterioritat si les necessita. El petit '*inconvenient*' és que un cop tancades, per tornar-les a activar, l'usuari ha de deixar allò que està fent. Si està programant una funció, si està utilitzant la finestra de comandes '*Command Window*' per fer càlculs, assignar variables, cridar funcions... pot resultar una mica '*molest*' aturar per encomanar-li al cap la tasca de saber com s'activa la finestra en qüestió que havia tancat abans i apartar les mans del teclat per dirigir-se a la barra de menú, prémer sobre '*view*' (o vista) i encertar en el desplegable que s'obre a continuació per activar de nou la finestra. Això és evitable si ens enrecordem de la comanda exacta per tal de tornar a activar les finestres per teclat, en tal cas el que estaríem fent és escriure una comanda en la finestra de comandes '*Command Window*' que es barreja amb les anteriors comandes que havíem escrit i que potser ens distreu de l'ús que estàvem fent d'ella.

D'altra banda hem comentat que a través de la finestra de comandes es pot sol·licitar ajuda d'una funció concreta i a la mateixa finestra de comandes '*Command Window*' se'ns mostra el contingut de l'ajuda. Això que és de gran utilitat, de vegades passa com en les situacions abans esmentades. Pot arribar a resultar molest escriure una sol·licitud d'ajuda a la mateixa finestra de comandes on estem treballant. Fins i tot de vegades el text corresponent a l'explicació d'ajuda pot arribar a ser extens, fins ocupar tota la pantalla de la finestra de comandes i deixar molt enrera les altres comandes que havíem escrit obligant-nos a anar enrera fent ús de la barra de desplaçament i en definitiva '*apartar-nos*' massa de la tasca que estàvem realitzant.

Pot haver més d'una comanda o funció de Matlab que no ens sigui fàcil de memoritzar ni d'escriure correctament per tal de cridar-la. Resultaria interessant poder cridar a aquella funció de la manera que ens sigui més fàcil i fins i tot tenir la opció de poder canviar la manera de cridar-la més endavant.

Per totes aquestes raons i donat que Matlab és un programa amb el qual es poden desenvolupar funcions i complements, va resultar interessant implementar un complement que dotés a Matlab o millor dit a l'usuari de Matlab d'un augment en les possibilitats de relacionar-se amb certes tasques de l'entorn del programari. Implementant a Matlab un sistema de reconeixement de la parla que serveixi per distingir diferents comandes podria ser una solució interessant. Aquest complement permetria que aquestes comandes puguin ser accionades de manera senzilla sense haver de deixar de fer la tasca principal que ens ocupa, no distreure'ns la vista ni ocupar la finestra de comandes amb altres accions, ni haver de preocupar-se en pensar com accionar una o altre. D'altra banda Matlab ens proporcionaria una proba de fins a on de fiable pot resultar el sistema de reconeixement de la parla escollit per possibles aplicacions en altres entorns.

2.2. ParLab, la necessitat d'un nom

Tal com s'ha comentat en l'apartat anterior, Matlab és un programa que permet desenvolupar funcions utilitzant un llenguatge. També proporciona un entorn gràfic on es pot dissenyar tota una interfície per facilitar la interacció. El propòsit de la creació d'un com-

plement de Matlab és precisament facilitar la interacció amb certs aspectes del programa. De res serviria fer un complement que a la pràctica resulti difícil de fer servir o de cridar. Com ja hem comentat amb anterioritat, les comandes a Matlab s'acostumen a realitzar a la finestra de comandes '*Command Window*'. Serà dons a través d'aquesta finestra com cridarem al nostre complement per tal de fer-lo servir. Per tant per poder fer la petició del complement, aquest necessita un nom.

Matlab és el nom del programa que neix com a conjunció de les primeres lletres de les paraules Matrix Laboratori (laboratori de matrius), fent així esment de la principal tasca que duu a terme i per la qual va ser creat.

Amb la mateixa filosofia, s'ha arribat a la conclusió que aquest complement, tracta del reconeixement de la parla a mode de laboratori. Amb la mateixa idea, i donat que es tracta d'un complement per aquest programa, s'ha configurat el nom mitjançant la conjunció de les primeres lletres de les paraules Parla Laboratori, naixent així el complement anomenat com **'ParLab'**. Al mateix temps la darrera síl·laba de la paraula 'parla' coincideix també amb la primera de la paraula 'laboratori', amb la qual cosa la paraula '*parla*' és sencera en el nom. A més a més el nom del complement 'ParLab' té el mateix número de lletres que el programari 'Matlab' i tan sols varia en dues de elles, la qual cosa el fa molt identificable amb el programari al qual serveix. Daltra banda, funcionalment és un nom curt fàcil de recordar i que repeteix lletres, cosa que el fa fàcil d'accionar en escriure'l ràpidament.

2.3. Components i funcions

Tot seguint els principis que han originat la implementació del complement 'ParLab', s'ha arribat a la conclusió que hi ha cinc tasques de l'entorn de Matlab que seria interessant poder accionar o interactuar amb elles mitjançant la parla. Aquestes tasques serien:

- Ajuda
- Neteja
- Workspace
- Directori
- Preferències

2.3.1. Ajuda

Es pot obtenir ajuda d'una funció de Matlab tot citant '`help`' seguit del nom de la funció en la finestra de comandes 'Command Window' tal com hem comentat a la secció 2.1.. No obstant hi ha una altra comanda que obre una finestra d'ajuda apart deixant lliure la finestra de comandes, tal com mostra la figura 2.3. D'aquesta manera es pot tenir disponible la finestra d'ajuda amb el text i fer noves cerques d'ajuda i tenir el 'Command

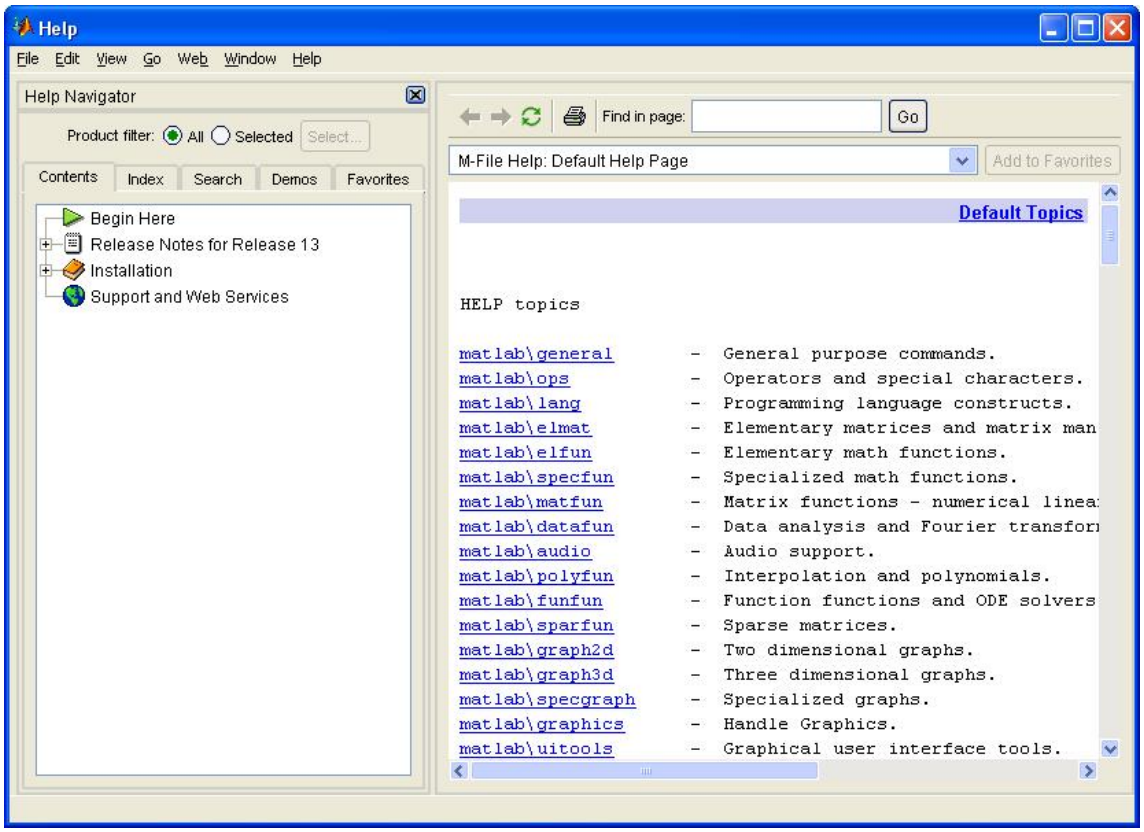

Figura 2.3: Finestra d'ajuda helpwin

Window' disponible amb el que havíem escrit abans o bé lliure per fer proves sense perdre de vista el text d'ajuda. Aquesta finestra d'ajuda s'acciona mitjançant la comanda '`helpwin`'.Accionar aquesta finestra seria d'utilitat però ja posats sense haver d'escriure la sentència a la '*Command Window*' i ni tan sols amb la necessitat de recordar-se del nom exacte per tal d'activar-la.

2.3.2. Neteja

De tant en tant la finestra de comandes '*Command Window*' està molt plena de text o operacions que ens interessa '*netejar*'. Aquesta és la comanda '`home`'. Fer una neteja de pantalla per comandament de veu serà altre de les funcions que incorporarà 'ParLab'.

2.3.3. Workspace

De vegades per aprofitar l'espai en pantalla, es pot optar per tancar la finestra del '*Workspace*', podent disposar així de tota la finestra del '*Command Window*' en la finestra de Matlab. Mitjançant la comanda '`Workspace`' es pot tornar a activar la finestra del '*Workspace*'. No obstant hem de fer una altra vegada ús de la finestra '*Command Window*' per fer la comanda, o bé activar-la per mitjà de la barra de menú, a la secció de vistes. En canvi per tancar la finestra '*Workspace*' amb una simple pulsació del ratolí la tanquem.

Donat que per desactivar aquesta finestra és molt més fàcil que per tornar-la a activar, aquesta serà doncs una altra funció que ens interessarà activar mitjançant la veu, així tant per tancar com per tornar-la a obrir bastarà una pulsació de ratolí.

2.3.4. Directori

Mitjançant aquesta comanda podrem veure tots els arxius i subdirectoris que tenim al directori de treball actual. Aquesta comanda es pot assolir a través del '*Command Window*' tot escrivint '`dir`'. Tot i tractar-se d'una comanda curta, cal escriure al '*Command Window*'. Resulta interessant quan hem tancat les finestres de '*Workspace*' i '*Current Directory*', poder visualitzar el contingut del directori.

2.3.5. Preferències

Si es vol accedir a la finestra de preferències del programari Matlab cal escriure a la finestra de '*Command Window*' la comanda '`preferences`'. Aquesta serà una altra funció que podrem accedir mitjançant una comanda parlada, sense fer ús de la finestra de comandes.

2.3.6. Calculadora, més d'un reconeixement

Acabem de veure cinc tasques que es podrien dur a terme sota comandaments de veu. Això implica que el nostre sistema reconeixedor de paraules hauria de poder distingir de l'ordre d'unes cinc paraules. És a dir, es pronuncia una paraula i el sistema ha de distingir d'entre una base de cinc paraules quina és la que se sembla més a la que hem pronunciat. Un cop té la paraula escollida, realitza la funció assignada. Però anem més enllà.

Què passaria si ampliem a una funció més. El sistema hauria de ser capaç de distingir sis paraules. I si a més a més aquesta nova funció es tractés d'una funció que espera de noves comandes que també ha de reconèixer per mitjà de la parla. Es tractaria doncs d'entrar en un submenú on s'esperen noves comandes per veu. Aquest desafiament d'anar més enllà és el que va motivar a fer la Calculadora parlada.

S'amplia així el sistema a una funció més que es dirà '**Calculadora**'. Així doncs el sistema disposarà de les següents funcions i paraules a reconèixer:

- Ajuda
- Neteja
- Workspace
- Directori
- Preferències

- Calculadora

Donades les característiques de la funció calculadora (la qual també serà parlada) el sistema s'amplia d'una manera especial.

En una primera fase, el sistema entra en un primer reconeixement. Entra una paraula nova al sistema pronunciada per nosaltres. El sistema encara no la ha reconegut. Per tant en aquesta primera fase farà un reconeixement de la paraula entrant tot comparant-la amb les sis paraules que configuren la base de paraules del sistema, tal com feia fins ara. Un cop reconeix la paraula, si es tracta d'una de les cinc paraules anteriors, el sistema executarà la comanda que tenia assignada com abans i haurà terminat. Si el sistema reconeix la paraula '*Calculadora*', el sistema entra en una segona i nova fase. Ara es tracta d'anar entrant els diferents paràmetres de la calculadora. Aquests paràmetres per ordre són:

1. Operació a realitzar
2. Allò que intervé en la operació: números o bé l'anterior resultat
3. Els números un a un

Ara ja no només es tracta d'una paraula nova que entra al sistema i que ha de ser identificada. Ara tenim moltes més paraules a identificar i de manera que ha de saber distingir cada paraula a quin 'camp' correspon d'identificar, si bé es tracta d'identificar una operació o bé es tracta d'un número o bé un mode d'operació.

El primer problema que tenim és que són massa paraules per identificar. D'altra banda es pot pensar que faria falta una base de paraules al sistema massa gran i el nostre sistema està pensat per bases de poques paraules. Però tenim una dada al nostre favor. La calculadora també té les seves instruccions. Si la manera i l'ordre de dictar els paràmetres sempre és la mateixa, el sistema sabrà que la primera paraula que dictem sempre fa referència a un conjunt de coses (tipus d'operació). La segona paraula fa referència a un altre conjunt de coses que és diferent (mode d'operació). La tercera paraula pertany a un altre conjunt de coses (els números).

Per tant ara el problema queda reduït a tres conjunts de paraules. Quan entra la primera paraula de la calculadora, haurà de fer un reconeixement dintre del primer conjunt: '*Operació*'. La segona paraula, haurà de fer un reconeixement dintre del segon conjunt: '*Memòria*'. A partir de la tercera paraula haurà de fer reconeixement dintre del tercer conjunt: '*Números*'.

Cada un d'aquests conjunts configuren una base de paraules. Per tant el sistema s'ha vist ampliat en tres bases de paraules noves que són exclusives de la funció calculadora. Per tant a més a més de la base de les diferents funcions i que hem afegit la funció de Calculadora, al sistema se li afegeixen les bases de operacions, memòria i números. Per tant el sistema disposarà de les següents bases:

Base de les diferents **funcions**:

- Ajuda
- Neteja
- Workspace
- Directori
- Preferències
- Calculadora

Base que configura les diferents **operacions**:

- suma
- resta
- multiplica
- divideix
- potencia

Base que configura la **memòria**:

- Números
- Resultat

Base que configura els diferents **números**:

- Zero
- Un
- Dos
- Tres
- Quatre
- Cinc
- Sis
- Set
- Vuit
- Nou

- Negatiu
- Coma
- Composició

Més endavant entrarem a parlar de quins mecanismes ha de seguir la calculadora per arribar a resoldre les operacions.

2.4. Interfície ParLab

Un dels motius que ha dut a la creació d'aquest complement per 'Matlab' és el facilitar la manera d'interactuar amb el propi programa. Seria una contradicció doncs, realitzar un complement que sigui més difícil de fer servir que el propi programa. Acabem de veure a la secció 2.3.6. que el complement o sistema 'ParLab' ha quedat 'ampliat' a diferents bases, cadascuna d'elles està composada per diferents paraules. Cal tenir en compte que en un sistema de reconeixement de la parla, un aspecte fonamental és la fase d'entrenament del sistema. Aquest punt és indispensable per un reconeixement òptim, ja que depenent del locutor unes paraules poden variar en el mode en què es pronuncien i si aquestes difereixen respecte les paraules que conformen la base pot afectar al correcte reconeixement. Fins i tot un mateix locutor difícilment pronunciarà una mateixa paraula exactament igual dues vegades. Per tant, es pot donar el cas que alguna paraula doni problemes perque s'ha guardat a la base amb una pronunciació diferent de l'habitual, però la resta de paraules no presentar problemes. També podria donar-se el cas de voler cambiar la manera en que anomenem una funció. És a dir voler canviar expressament la paraula pronunciada que activa un comandament, per anomenar aquest comandament d'una altra manera. En aquests casos convindria de disposar d'un mètode per corregir una sola paraula de la base de paraules sense tornar a gravar les altres.

Per tant veiem que el complement 'ParLab' requereix de flexibilitat alhora fer-lo servir. Aquesta flexibilitat implica un entrenament de tot el sistema i la posibilitat de correcció de parts del sistema per millorar l'entrenament. Tot això cal que es pugui fer de manera senzilla i sense possibilitat d'error. Quan l'usuari assigna una paraula a una funció no té perquè saber de memòria quin és el nom de la paraula a pronunciar ni molt menys hauria de molestar-se en escriure lletres per fer esment de la paraula que vol configurar ja que a més a més d'ésser molt incòmode, incrementaria la posibilitat d'errors en l'escriptura i en l'execució del sistema. Daltra banda el complement hauria de ser el més compacte possible, ja que es tracta d'això, un complement a Matlab, ocupant poc en pantalla donat que poseeix funcions per activar finestres que suposadament s'han tancat per possibles problemes d'espai o per millorar la visibilitat i el fet contrari també esdevindria contradictori.

Per tant amb aquests quatre requisits:

- Facilitat d'ús
- Flexibilitat de configuració

- No posibilitat d'error alhora de fer-lo servir
- Tamany compacte

s'han establert els criteris de disseny de la interfície de 'ParLab'.

Mitjançant una plataforma 'GUI' s'ha dissenyat la interfície que mostra la figura 2.4. El

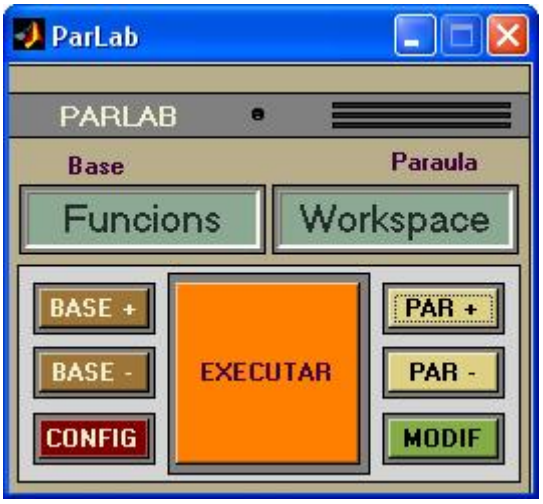

Figura 2.4: Interfície ParLab

conjunt està format per set botons i dues pantalles a mode de display de cristall líquid. La filosofia d'aquest complement és que sigui el menys 'visual' possible i el màxim sonor possible. És a dir, la idea és apropar-se al màxim al món 'àudio' fugint del món 'visual'. Interactuar amb la parla implica parlar i escoltar, orientat així al sentit de la oïda, diferent del visual. La idea és ajudar a interactuar amb el programa per una altra via diferent. Bé perquè el sentit de la vista està ocupat en la pantalla en altres tasques o bé perquè una de les aplicacions dels sistemes de reconeixement de la parla més enllà del propi programa sigui el possible desenvolupament de sistemes d'ajut a persones invidents. En aquesta línia es podria haver dissenyat la interfície encara més compacta, tot eliminant la part superior fins arribar a les pantalles, tot i que s'ha estimat més fer constar el nom del complement 'ParLab' i una decoració que recordi a un petit micròfon en la part central i unes escletxes que farien les vegades de petit altaveu a la part superior dreta, tot fent recordar que es tracta d'un sistema que necessita micròfon i altaveu. D'altra banda també recorda que en un futur altres funcions com la calculadora de ser millorades podrien incorporar-se en dispositius portàtils per ajudar a gent que així ho precisi. També es va pensar en un principi no dotar de pantalles al sistema i proporcionar tota la informació en missatges sonors parlats. Finalment es va considerar que pel nostre complement per Matlab les pantalles tenien la seva aplicació proporcionant més velocitat de maniobrabilitat per l'entrenament alhora que servia per visualitzar les diferents operacions de la calculadora.

2.4.1. Comandaments

La interfície de 'ParLab' consta d'elements clarament diferenciables. En un primer cop d'ull a la figura 2.4 podem apreciar dos elements clarament diferenciats: pantalles i botons.

Si parem atenció en com estan disposades les pantalles i els botons, veurem que les pantalles estan una al costat de l'altre i que sota de cada pantalla hi ha una sèrie de botons disposats en columna. Així doncs tenim dues columnes de botons, una a sota de cada pantalla i separades per un botó central i mes gran; aquest és el botó de 'Executar'. A sobre de les pantalles hi ha unes lletres tot indicant l'ús que es fa d'aquella pantalla i de tots els botons en columna sota d'ella. Per tant tenim que a la columna de l'esquerra es disposen els comandaments que fan referència a les 'Bases' i a la columna de la dreta els que fan referència a les 'Paraules'. Per tant amb els botons de l'esquerra, ('BASE +' o 'BASE -') seleccionarem la base tot visualitzant-la a la pantalla superior esquerra per a configurar-la mitjançant el botó 'CONFIG' i amb els botons de la dreta ('PAR +' o 'PAR -') seleccionarem la paraula que vulguem modificar tot visualitzant-la a la pantalla superior dreta per a modificar-la mitjançant el botó 'MODIF'. Les pantalles també seran utilitzades per la calculadora per mostrar els números que intervenen en el càlcul, el tipus d'operació i el resultat.

Per tant podem dir que els comandaments formats pels botons en columna a esquerra i dreta juntament amb les pantalles serveixen per l'*entrenament* del sistema, mentre que el botó central més gran serveix per *executar* el sistema de reconeixement de paraules.

2.4.1.1. Botó 'BASE +'

Aquest botó serveix per especificar quina base volem configurar o la base de la qual volem modificar alguna paraula. Al prémer aquest botó visualitzarem el nom de la base a la pantalla de bases situada a la part superior esquerra i automàticament es mostrarà el nom de la primera paraula continguda a la base en la pantalla de paraules situada a la part superior dreta. Si volem especificar altra base diferent, només hem de prémer una altra vegada el botó per que se'ns mostri la següent base. Les diferents bases es van mostrant a la pantalla de forma seqüencial. Al tractar-se del botó 'BASE **+**' l'ordre d'aparició de les diferents bases és ascendent, és a dir, al prémer la primera vegada visualitzarem la primera base, al prémer una altra vegada veurem la segona base i així successivament. Quan arribi al final es tornarà a començar per la primera. Si es vol modificar alguna paraula que sigui d'una base diferent, primer cal escollir la base tot utilitzant aquest botó.

Donat que aquesta aplicació és pensada per interactuar mitjançant la parla, i sentint comandaments amb l'oïda amb la filosofia de ser el "menys visual possible", va resultar interessant dotar de diferents sons les diferents bases per així poder distingir mitjançant la oïda a quina base ens trobem. Amb aquest objectiu, cada base té associada una nota musical tot seguint una escala major. D'aquesta manera, la primera base tindrà la nota tònica de la escala (per exemple un Do en l'escala de Do Major) i a mesura que augmentem el desplaçament de base augmentarà el so de la nota desplaçant-nos així per l'escala musical.

2.4.1.2. Botó 'BASE -'

Es tracta d'un botó anàleg al botó 'BASE +', però en aquest cas, com que es tracta del botó 'BASE **-**' l'ordre d'aparició per pantalla de les diferents bases és descendent. Anàlogament

al botó 'BASE +' les bases tenen associat un so corresponent a una nota musical, i a mesura que seleccionem les diferents bases de forma descendent el so també anirà canviant desplaçant-nos així per l'escala musical de forma descendent. Així doncs els botons 'BASE +' i 'BASE -' es complementen l'un a l'altre de tal manera que utilitzarem un o altre per seleccionar la base que ens interessi de forma més ràpida i no haver d'esperar a "donar tota la volta" a la seqüència de bases en el cas d'utilitzar tan sols un botó de tots dos.

2.4.1.3. Botó 'PAR +'

De vegades és necessari tornar a gravar o corregir una sola paraula del sistema. Per tal de no tornar a gravar totes les paraules d'una base de nou, és útil disposar d'una opció senzilla d'utilitzar, pràctica i alhora que eviti errors en el nom de la paraula a gravar.

Aquest botó serveix per visualitzar les diferents paraules que componen una base i per especificar quina paraula del sistema es vol tornar a gravar i processar. Cada vegada que es premi aquest botó es mostrarà una paraula diferent de la base. Les paraules es van mostrant a la pantalla de paraules situada a la part superior dreta. Al prémer de manera successiva el botó es van visualitzant les diferents paraules de forma ordenada i seqüencial corresponents a la base que tenim seleccionada. Al tractar-se del botó 'PAR **+**' l'ordre d'aparició de les paraules és ascendent, és a dir, al prémer la primera vegada visualitzarem la segona paraula, al prémer una altra vegada veurem la tercera paraula i així successivament. Quan arribi al final de les paraules que conformen la base, tornarà a començar per la primera de la mateixa base. Si volem arribar a una paraula que està en una altra base, primer haurem d'especificar la base que conté la paraula en qüestió fent servir els botons 'BASE +' o 'BASE -'. Al canviar de base se'ns mostrarà automàticament el nom de la primera paraula continguda a la nova base que hem seleccionat. Per tant si es tracta de configurar la primera paraula no caldrà prémer cap més botó.

Donat que aquesta aplicació és pensada per interactuar mitjançant la parla, i sentint comandaments amb l'oïda amb la filosofia de ser el "menys visual possible", va resultar interessant dotar de diferents sons les diferents paraules per així poder distingir mitjançant la oïda a quina paraula de la base ens trobem, de la mateixa manera que amb el botó 'BASE +'. Amb aquest objectiu, cada paraula té associada una nota musical tot seguint una escala major. D'aquesta manera, la primera paraula de la base tindrà la nota tònica de la escala (per exemple un Do en l'escala de Do Major) i a mesura que augmentem el desplaçament de paraules en la base augmentarà el so de la nota desplaçant-nos així per l'escala musical.

2.4.1.4. Botó 'PAR -'

Es tracta d'un botó anàleg al botó 'PAR +', però en aquest cas, com que es tracta del botó 'PAR **-**' l'ordre d'aparició per pantalla de les diferents paraules és descendent. Anàlogament al botó 'PAR +' les paraules tenen associat un so corresponent a una nota musical, i a mesura que seleccionem les diferents paraules de forma descendent el so també anirà canviant desplaçant-nos així per l'escala musical de forma descendent. Així doncs els botons 'PAR +' i 'PAR -' es complementen l'un a l'altre de tal manera que utilitzarem un o

altre per seleccionar la paraula que ens interessi de forma més ràpida i no haver d'esperar a "donar tota la volta" a la seqüència de les paraules d'una base en el cas d'utilitzar tan sols un botó de tots dos.

2.4.1.5. Botó 'CONFIG'

Es tracta del botó per configurar la base seleccionada. Primer hem de seleccionar la base que volem configurar amb els botons "BASE +" o "BASE -". Mitjançant aquests botons podem visualitzar les diferents bases del sistema a la caixa de text 'Base' o bé guiarnos pel so distintiu que emet cada base. A continuació podem prémer el botó "CONFIG" per configurar-la. Un cop es prem el botó "CONFIG" es procedeix a la gravació de cada paraula de la base. Un so de mig segon de durada ens avisa just abans de cada gravació. Immediatament després de sentir aquest so, tenim uns tres segons per procedir a gravar una paraula de la base mitjançant el micròfon. Transcorregut aquest temps, si tot va bé, tornarem a sentir una altra vegada el so que ens indica que podem enregistrar la següent paraula de la base. Un cop es finalitza l'enregistrament de totes les paraules de la base, un so més agut ens indica que s'han gravat totes les paraules. A continuació es processa tota la base automàticament i un missatge sonor ens diu que el sistema està configurat.

Al enregistrar cada paraula, es verifica que es compleixen els criteris de qualitat del senyal definits pel sistema. Si no es compleixen, el sistema ens avisa mitjançant un missatge sonor i ens 'obliga' a tornar a gravar la mateixa paraula fins que es compleixin aquests criteris.

Cal tenir en compte que al gravar una paraula es genera un pic que s'eliminarà al enregistrar la paraula mitjançant la funció '`grava`' per tal de tenir una gravació més "neta" per procedir al processament de la paraula. Tal com s'explica en la funció '`grava`' s'eliminaran les 2000 primeres mostres de la gravació, això equival a dir que amb una freqüència de mostreig de 8000 mostres per segon s'eliminaran 0,25 segons. Per tant hem de tenir en compte que quan sona el senyal de gravació de la següent paraula, en realitat la gravació que ens interessa començarà un quart de segon després ja que tot so enregistrat durant aquest quart de segon quedarà automàticament eliminat. D'altra banda, aquest quart de segon s'estima que és un temps raonable de no producció de so donat que en sentir el so que ens avisa per començar a pronunciar la següent paraula hi ha un temps de reacció a aquest estímul i un temps per agafar aire per començar a parlar. S'estima que aquest temps de reacció i el temps per agafar aire supera les 25 centèsimes de segon.

2.4.1.6. Botó 'MODIF'

Tal com s'ha comentat explicant el funcionament dels botons 'PAR +' i 'PAR -', de vegades és útil i necessari corregir o tornar a enregistrar una sola paraula del sistema. Un cop s'ha especificat la paraula que volem corregir mitjançant els botons 'PAR +' o 'PAR -' de la base escollida mitjançant els botons 'BASE +' o 'BASE -', podem procedir al seu enregistrament mitjançant el botó 'MODIF'. Al prémer aquest botó, un missatge sonor ens anuncia que hem entrat en la opció de modificar el sistema i ens dicta les instruccions a seguir. A continuació un so de mig segon de durada ens avisa del principi de la gravació. Tot seguit

disposem d'uns tres segons per pronunciar la paraula a través del micròfon. El sistema eliminarà el 'pic' del principi de la gravació i farà una mesura dels criteris de qualitat del senyal del sistema. Si no es compleixen els criteris de qualitat del senyal, el sistema ens mostrarà un missatge sonor d'error tot indicant-nos la fallida de la qualitat del senyal i 'obligant-nos' a tornar a enregistrar la paraula fins que es compleixin aquests criteris de qualitat. Un cop tenim la paraula enregistrada complint els criteris mínims de qualitat es tornarà a processar automàticament tota la base incorporant aquesta darrera paraula ja modificada. Finalment, un missatge sonor ens indicarà que el sistema està modificat.

2.4.1.7. Botó 'EXECUTAR'

Mitjançant aquest botó, es posa en funcionament el sistema de reconeixement de paraules. Un cop tinguem el sistema 'entrenat', podrem executar les diferents funcions mitjançant la parla després de prémer el botó. Quan vulguem executar una funció determinada, primer premem el botó 'executar', seguidament sonarà un to de mig segon de durada que ens avisa del començament de l'enregistrament d'una paraula. A continuació disposem d'uns 3 segons per pronunciar a través del micròfon una de les paraules corresponent a la funció que desitgem executar. Aquesta paraula, un cop enregistrada, es prepara en condicions de ser processada. Aquest acondicionament previ consta de dues etapes. En una primera etapa, es talla el pic que es genera al accionar el sistema de gravació. Aquest pic s'estima que ja no existeix més enllà de les primeres 2000 mostres. Per tant es tallaran les 2000 primeres mostres enregistrades. La segona etapa consta del control de la qualitat del senyal. Es comprova que la paraula pronunciada supera els criteris de qualitat proporcionats pel sistema. Si la paraula no supera aquest control de qualitat, el sistema emet un missatge sonor tot indicant l'error de qualitat del senyal i ens obliga a tornar a pronunciar la paraula en millors condicions. Un cop superades aquestes dues etapes, es pot considerar que el sistema disposa d'una paraula entrant que cal que sigui identificada. Tot seguit el sistema posa en marxa automàticament el mecanisme de reconeixement de la paraula que hem pronunciat, tot comparant-la amb les paraules que es disposen a la base. Un cop el sistema identifiqui la paraula, la funció identificada serà anunciada mitjançant un missatge sonor i a continuació s'executarà automàticament la funció corresponent.

Les possibles funcions a identificar, i que per tant pronunciarem, són:

Calculadora, Neteja, Workspace, Directori, Preferències, Ajuda.

Al reconèixer la paraula '*Neteja*', la pantalla de comandes 'Command Window' del Matlab quedarà neta amb el cursor situat a la part de dalt.

Al reconèixer la paraula '*Workspace*' s'activarà la finestra del 'workspace' de Matlab si no la teníem activada.

Al reconèixer la paraula '*Directori*' ens mostrarà el directori actual.

Al reconèixer la paraula '*Preferències*' s'activarà la finestra de preferències del programa Matlab.

Al reconèixer la paraula '*Ajuda*' s'activarà la finestra d'ajuda del Matlab, per tal de buscar

els temes d'ajuda en una finestra apart i no utilitzar així la finestra de comandes 'command window'.

En el cas de pronunciar la paraula '*Calculadora*', el sistema al reconèixer aquesta paraula emetrà un missatge sonor per indicar que ha reconegut la paraula i que inicia la funció 'Calculadora' la qual és una mica diferent a la resta de funcions. Concretament se sentirà el missatge "*Iniciant calculadora*" i tot seguit entrem en l'inici dels paràmetres de la calculadora, els quals s'accionaran també mitjançant la veu. Entrem així en un segon sistema o subsistema de reconeixement de paraules. Aquestes paraules seran els comandaments que cal introduir per ordre per fer funcionar la calculadora.

La calculadora pot fer operacions de suma, resta, multiplicació, divisió i potència. La calculadora està dissenyada per poder operar amb números positius i negatius de fins a 9 decimals. El càlcul només és d'un tipus d'operació cada vegada que s'activa la calculadora. En principi, està pensat que en la operació a realitzar puguin intervenir tants números com desitgem, és a dir, podríem, per exemple, sumar d'una vegada 2, 3, 10 o 24 xifres, tot i que a la pràctica disposarem de limitacions del maquinari. En una següent activació de la calculadora, es pot realitzar una operació tot tenint en compte l'últim resultat, és a dir, el resultat de l'anterior vegada que es va activar. Permet, per tant, una acumulació de resultat. Això és possible amb la paraula '*resultat*'. Mitjançant la paraula '*resultat*', el darrer resultat s'incorporarà a la operació com si fos un número més a intervenir en el càlcul. Si no volem incorporar el resultat, haurem de pronunciar la paraula '*números*' per indicar-li a la calculadora que només volem operar amb els números que li dictarem.

Com que la calculadora necessita més d'un paràmetre, és necessari l'enregistrament de totes aquestes paraules corresponents als diferents paràmetres per tal que posteriorment el sistema pugui fer un reconeixement de cadascun d'ells. Per tant, seguidament al missatge sonor "*iniciant calculadora*" s'inicia una sèrie d'enregistraments de paraules. Aquests enregistraments s'inicien amb un to de mig segon de durada que ens marca l'inici de l'enregistrament d'una paraula. A partir d'aquest to disposem d'uns 3 segons per enregistrar una paraula. Seguidament el sistema fa el control de qualitat del senyal i si el supera emet un altre to per procedir a l'enregistrament de la següent paraula. Com que la calculadora no sap a priori quants números i de quin tipus (positius, negatius, decimals, número de xifres que els componen) intervenen en el càlcul, el sistema continuarà enregistrant paraules una darrera altra sempre que es superin els criteris de qualitat del senyal. Per indicar que s'ha arribat al final de l'entrada de paràmetres per la calculadora i que ja no necessitem continuar enregistrant paraules, a la següent paraula que toqui enregistrar simplement no hem de pronunciar res, i el silenci li comunicarà al sistema que hem acabat de entrar paràmetres i pararà d'enregistrar.

Aquests paràmetres de la calculadora per ordre són: operació a realitzar (*suma*, *resta*, *multiplica*, *divideix*, *potencia*), què intervindrà a la operació (escollim entre *números* o *resultat*), números que intervenen en la operació. Per números negatius, primer cal especificar el signe negatiu mitjançant la paraula '*negatiu*' i en el següent enregistrament pronunciar el número. Per entrar números de vàries xifres cal primer fer-li saber al sistema que el número que entrarem estarà *composat* per vàries xifres. Això es farà enregistrant la paraula '*composició*' i en el següent enregistrament pronunciar el número corresponent al número de xifres que componen el número a entrar i en els posteriors enregistraments pronunciarem una per una les xifres que formen el número. Així doncs per multiplicar 3

per -10 enregistrarem totes aquestes paraules: *multiplica*, *números*, *tres*, *negatiu* (perquè el número que ve a continuació és negatiu), *composició* (perquè el número que ve a continuació és de més d'una xifra), *dos* (perquè el número és de dues xifres), *un* (primera xifra del número), *zero* (segona xifra del número).

Si volem introduir un número decimal, primer enregistrem la part entera del número, seguidament indicarem que el número és decimal tot enregistrant la paraula '*coma*'. A continuació hem d'especificar el número de decimals que tindrà tot enregistrant un número corresponent al número de decimals. Finalment procedirem a l'enregistrament de les xifres decimals una per una. Per exemple si volem enregistrar el número decimal 3,14 haurem d'enregistrar les paraules: *tres*, *coma*, *dos* (perquè té dues xifres decimals), *un* (primera xifra decimal), *quatre* (segona xifra decimal).

a Exemple complert

Volem utilitzar ParLab per realitzar mitjançant la veu el càlcul:

$$3 * (-5,43) * (-11,57)$$

Passos a seguir:

1. Premem el botó '*Executar*' del ParLab
2. Després del senyal acústic pronunciem la paraula '*Calculadora*'
3. Si tot va bé, després de sentir la frase '*iniciant calculadora*' entrem a la funció calculadora i passem a enregistrar els paràmetres de la calculadora. Per tant després de cada senyal acústic pronunciarem una de les paraules i esperarem al següent senyal acústic per pronunciar la següent paraula.
4. Procedim a pronunciar les paraules següents una a una després de cada senyal acústic: *multiplica*, *números*, *tres*, *negatiu*, *cinc*, *coma*, *dos*, *quatre*, *tres*, *negatiu*, *composició*, *dos*, *un*, *un*, *coma*, *dos*, *cinc*, *set*
5. Després del següent senyal acústic no diem res per indicar que hem acabat d'entrar paràmetres.
6. Finalment si tot va bé, la calculadora mostrarà la operació i els números que han intervingut en la pantalla de 'base' i el resultat en la pantalla de 'paraules' i ens dictarà el resultat número per número. En aquest cas hauríem de sentir les paraules: *un*, *vuit*, *vuit*, *coma*, *quatre*, *set*, *cinc*, *tres*

2.5. Funcions associades als comandaments del sistema

Acabem de veure a la secció 2.4.1. una descripció dels diferents components i comandaments o botons que formen part de la interfície gràfica de 'ParLab'. Per tal que la interfície

gràfica resulti operativa i per tant 'cobri vida' cal que a cada comandament i element de la interfície hi hagi associades unes funcions. Aquest és el codi que està associat a la interfície gràfica (GUI) i que hem anomenat 'ParLab'. Daltra banda la interfície és una forma més 'còmoda' d'interactuar amb el nostre sistema de reconeixement de paraules. Significa això que primer hi ha tot el sistema de reconeixement de paraules el qual necessita tot un conjunt de funcions que fan possible que el sistema funcioni i per altra banda hi ha la interfície que permet posar en marxa i fer funcionar de manera fàcil el sistema. En termes de mecànica, parlar de les funcions associades al sistema de reconeixement seria com parlar del 'motor' i parlar de la interfície gràfica (GUI) seria com parlar del 'quadre de comandaments'. Quan tinguem en el nostre directori de treball el codi de la interfície gràfica 'ParLab' i totes les funcions associades als sistema de reconeixement i subdirectoris necessaris, teclejant a la finestra de comandes '*Command Window*' la instrucció:

```
>> parlab
```

s'activarà la interfície gràfica i aquesta podrà trobar totes les funcions necessàries per poder funcionar.

A continuació passarem a descriure totes les funcions que fan possible el sistema de reconeixement de paraules i com es relacionen entre elles i finalment la manera de fer-les servir a través de la interfície gràfica amb el codi associat. Finalment la figura 2.12 mostra un 'mapa' de totes les funcions i la dependència que hi ha entre elles.

2.5.1. Condicions

El nostre sistema de reconeixement precisa de tot un seguit d'operacions amb dades. Al tractar-se de reconeixement de la parla, haurà d'enregistrar paraules i aquestes paraules les haurà de processar per extreure unes característiques. Significa això que les paraules hauran estat enregistrades en format digital i, per tant, caldrà determinar una freqüència de mostreig. Cada enregistrament serà d'una certa durada de temps, per tant també caldrà especificar la durada dels respectius enregistraments. Les diferents operacions a realitzar amb les paraules un cop enregistrades (aquí ja podríem dir operacions amb mostres) inclouen enfinestrat (on caldrà saber l'amplada de la finestra i la separació entre finestres), extracció de coeficients Cepstrum (caldrà saber en número de coeficients que volem), coeficients Delta Cepstrum (on caldrà saber l'índex de coeficients Cepstrum a tenir en compte en el càlcul). D'altra banda el sistema de reconeixement de paraules ha de saber quantes paraules ha de poder distingir en una primera identificació (el número de paraules del sistema corresponents a una primera fase de reconeixement de les diferents tasques sense entrar en la calculadora). Les diferents paraules agrupades en les diferents bases. Totes aquestes dades que proporcionen les condicions del sistema és el que hem anomenat '*condicions*.

'*condicions*' és un arxiu que conté al seu interior diferents variables que seran utilitzades per diferents funcions en tot el sistema. Per generar aquest arxiu s'ha creat la funció també anomenada '`condicions.m`'. En la funció '`condicions.m`' s'han declarat les variables corresponents al sistema i finalment aquestes són guardades. Si es vol rectificar alguna variable del sistema caldrà obrir doncs la funció '`condicions.m`', rectificar l'assig-

nació de la variable a canviar, guardar la funció un cop canviada i a continuació cridar la funció a la finestra de comandes 'Command Window' per tal que aquesta generi l'arxiu '*condicions*' que contindrà les noves variables del sistema. A continuació detallem les variables definides a la funció '`condicions.m`'.

2.5.1.1. Freqüència de mostreig 'Fs'

El sistema tracta de reconeixement de paraules. Aquestes paraules seran pronunciades per un interlocutor mitjançant un micròfon per ser enregistrades per l'ordinador. La veu humana té un to fonamental mitjà que comprèn un rang de freqüències entre els 100 i els 300 Hz. La veu humana és modulada mitjançant el tracte vocal. A nivell de conversa les cordes vocals poden generar sons compresos entre els 250 y 3.000 Hz, tot i que alguns fonemes poden estar situats entre els 4000 i els 8000 Hz. Significa això que les freqüències agudes ajuden a identificar els fonemes. D'altra banda la capacitat dels éssers humans per distingir paraules i entendre una conversa demostra que no és necessari una font sonora que transmeti totes les freqüències, sinó un rang molt inferior. Els sistemes de telefonia arriben fins uns 4000 Hz, fet que fa que no es transmeti la veu exactament igual, sentint-se una mica distorsionada, però en qualsevol cas és suficient perquè es pugui arribar a entendre. Per tant tenint en compte l'espectre mitjà de la veu humana i que fins a uns 4000 Hz de la font sonora és més que suficient per garantir la distinció de paraules, agafem una freqüència màxima del senyal (F_{max}) inferior als 4000 Hz. El '*criteri de Nyquist*' ens diu que per garantir que un senyal sigui totalment reconstruible i evitar el solapament o '*aliasing*' cal que la '*freqüència de mostreig*' (F_s) sigui superior al doble de la màxima freqüència del senyal[5, pàg. 29][6, pàg. 307]:

$$F_s > 2 \cdot F_{max} \tag{2.1}$$

Per tant per garantir la correcta reconstrucció del senyal a analògic tot complint amb el criteri de Nyquist s'arriba a la conclusió que una freqüència de mostreig de **8000 mostres per segon** és suficient.

Si atenem al fet que les freqüències agudes ajuden a distingir els fonemes, una freqüència de mostreig de 8000 mostres per segon implica que els sons propers a 4000 Hz tindrien una representació de dues mostres per cicle i per tant alguns fonemes resultarien un tant limitats. Si a més a més alguns fonemes presenten freqüències situades entre els 4000 i els 8000 Hz, aquestes superen la freqüència màxima i per tant no estarien representades. Per aquest fet fem notar que cal trobar un equilibri entre la 'bona' representació del senyal i la 'òptima' per tal que pugui ser intel·ligible pel sistema i alhora li proporcioni un millor rendiment computacional.

S'ha considerat que una freqüència de mostreig de 8000 mostres per segon proporciona un rendiment acceptable pel reconeixement per part del sistema de les sis paraules que configuren les sis tasques principals dels comandaments per veu i un pes i rendiment computacional adequat.

2.5.1.2. Temps de durada d'una paraula 'tempsParaula'

Aquest és el temps en segons que durarà la gravació d'una paraula. El temps es guarda a la variable `tempsParaula` i el seu valor és de **3 segons**.

2.5.1.3. Amplada de Finestra 'AF'

El sistema de reconeixement de paraules necessita extreure unes característiques de cada paraula, tant alhora d'entrenar el sistema com quan entra una paraula per ser identificada. Aquestes característiques inclouen una segmentació en 'trames' per aconseguir tenir en una trama una porció de la paraula de longitud propera a un fonema, per poder després aplicar un tractament de Cepstrum a cada trama. Aquesta trama es construeix per mitjà d'un enfinestrat del senyal. L'amplada de la trama, és a dir de la finestra, haurà de ser propera a la durada d'un fonema. S'estima que aquesta amplada pot tenir una durada mitja aproximada d'uns **20 mil·lisegons**.

S'ha definit una variable 'AF' (*Amplada Finestra*' que representa l'amplada d'aquesta finestra en segons i que es guardarà amb el valor assignat de 20e-3 (és a dir 20 mil·lisegons).

2.5.1.4. Separació entre Finestres 'SepF'

Acabem de veure que cada paraula serà segmentada en trames o finestres d'uns 20 mil·lisegons. A cada 'trama' se li extrauran unes característiques basades en el càlcul dels coeficients Cepstrum que després s'ampliaran amb els coeficients Delta Cepstrum. Hem vist a la part teòrica que degut a les operacions del Cepstrum, aquesta 'trama' no és adient de ser construïda a partir d'un enfinestrat rectangular. També hem vist que un enfinestrat òptim per a sistemes de reconeixement basats en Cepstrum és el de tipus 'Hamming' i que degut a les característiques de l'enfinestrat Hamming les trames necessiten un cert '*solapament*' entre finestres. Aquest 'solapament' el determina la separació entre finestres. Hem vist a la part teòrica que en finestres Hamming els extrems de la finestra presenten atenuació, i que per compensar aquesta atenuació, s'acostuma a fer coincidir el màxim de la següent finestra (el centre de la finestra) amb l'extrem de la finestra anterior. Per tant per una amplada de finestra de 20 mil·lisegons es considera òptima una separació entre finestres de **10 mil·lisegons**.

S'ha definit la variable 'SepF' (*Separació Finestra*) que representa la separació entre finestres en segons i que es guardarà amb el valor assignat de 10e-3 (és a dir 10 mil·lisegons).

2.5.1.5. Número de Coeficients Cepstrum 'NC'

De cada trama o finestra cal extreure unes característiques. Aquestes característiques parteixen del càlcul del Cepstrum i la selecció d'un número de coeficients. Com hem vist a la part teòrica es considera que per un sistema de reconeixement de paraules la informació que interessa està continguda en els primers coeficients del Cepstrum els quals

proporcionen les característiques del tracte vocal. S'estima que es pot tenir una representació òptima del tracte vocal amb uns **10 coeficients**.

S'ha definit la variable 'NC' (*Número coeficients Cepstrum*) que representa el número de coeficients del cepstrum que farà servir el sistema i que es guardarà amb el valor assignat de 10.

2.5.1.6. Índex de coeficients Cepstrum 'ND'

Un cop s'han extret els coeficients Cepstrum, hem vist que s'acostuma a acompanyar a aquests coeficients amb uns coeficients anomenats '*Delta-Cepstrum*' i que es calculen tenint en compte una sèrie de coeficients Cepstrum. Aquests nous coeficients 'Delta-Cepstrum' juntament amb els coeficients Cepstrum formaran les característiques.

En el sistema s'ha declarat una variable anomenada 'ND' que correspon a l'**índex** de coeficients Cepstrum a tenir en compte en el càlcul d'un coeficient Delta-Cepstrum. A aquesta variable es guardarà amb un valor assignat de **2**. El número de coeficients serà 2*ND+1.

2.5.1.7. Número de paraules del sistema 'nPS'

El nostre sistema de reconeixement de paraules està orientat cap a la possibilitat d'execució de tasques mitjançant un sol comandament, la parla. Depenent de la paraula que es pronunciï, el sistema mitjançant el seu reconeixement executarà una tasca o una altra. A priori el número de tasques a poder executar correspon a les **sis** funcions principals que executarà el sistema en un primer reconeixement. Recordem que aquestes sis funcions i, per tant, sis paraules són: *'Neteja', 'Workspace', 'Directori', 'Preferències', 'Ajuda' i 'Calculadora'*. Per tant el sistema ha de poder reconèixer una paraula entre aquestes sis paraules. Hem vist que degut a la tasca '*calculadora*' el sistema s'amplia a una segona fase de reconeixement i, per tant, el número de paraules a reconèixer queda ampliat amb altres '*conjunts*' de paraules agrupades en diferents *bases*. Però aquesta segona fase de reconeixement d'aquestes paraules formen part única i exclusivament de la tasca '*Calculadora*' a la qual cal haver entrat prèviament mitjançant el primer reconeixement. Per tant aquestes paraules agrupades en altres '*conjunts*' o bases no es contemplen aquest paràmetre.

S'ha definit la variable 'nPS' (que representa el *número de Paraules del Sistema*) i que es guardarà amb el valor assignat de **6**.

2.5.1.8. Bases

Hem vist a la secció 2.3.6. que degut al reconeixement de la tasca '*calculadora*' el sistema passa a una segona fase de reconeixement on requereix de més paraules. Necessita doncs ser ampliat amb uns '*conjunts*' de paraules que anomenem '*bases*'. A la funció 'condicions.m' s'ha definit un vector ('Bases') que té com a complements els noms dels diferents '*conjunts*' o '*bases*' del sistema i que es guardarà a l'arxiu '*condicions*'. El contingut d'aquest vector serà doncs:

'*Funcions*', '*Números*', '*Operacions*', '*Memòria*'.

2.5.1.9. Funcions

El conjunt de paraules relatiu a les sis paraules del sistema, d'entre les quals s'ha de poder reconèixer una d'elles cada vegada que volem executar una comanda és la base de paraules que hem anomenat '*Funcions*'. A la funció '`condicions.m`' s'ha definit un vector '*Funcions*' que té com a components els noms de les diferents paraules del sistema corresponents a les sis tasques a reconèixer i que es guardarà a l'arxiu '*condicions*'. El contingut d'aquest vector és:

'*Calculadora*', '*Neteja*', '*Workspace*', '*Directori*', '*Preferències*', '*Ajuda*'.

2.5.1.10. Operacions

Quan es vol activar el comandament '*calculadora*', premerem el botó '*executar*' i pronunciarem la paraula '*calculadora*'. El sistema ha de reconèixer la paraula i entrar en la tasca 'calculadora'. Hem vist que en aquest moment el sistema entra en una segona fase i requereix de nous paràmetres. Aquests paràmetres són exclusius de la calculadora i són noves paraules que s'agrupen en tres '*conjunts*' diferents donant origen a tres noves bases de paraules. Una d'aquestes bases correspon a les diferents operacions que pot realitzar la calculadora. Amb aquest propòsit, a la funció '`condicions.m`' s'ha definit un vector anomenat '*Operacions*' que es guardarà a l'arxiu '*condicions*' i que té com a components els noms de les diferents paraules necessàries per poder assignar les operacions a la calculadora. El contingut d'aquest vector és:

'*Suma*', '*Resta*', '*Multiplica*', '*Divideix*', '*Potencia*'.

2.5.1.11. Memòria

Una altre de les bases o '*conjunts*' de paraules que necessita la calculadora fa referència als elements que intervindran en el càlcul. Aquests elements poden ser '*números*' (farà els càlculs amb els números que li dictem) o bé '*Resultat*' (a més a més de fer el càlcul amb els números que dictem, afegirà el darrer resultat obtingut com un número més a calcular).

A la funció '`condicions.m`' s'ha definit un vector anomenat '*Memoria*' que es guardarà a l'arxiu '*condicions*' i que té com a components els noms de les paraules que fan possible l'elecció de la incorporació, o la no incorporació, del resultat. El contingut d'aquest vector és:

'*Numeros*', '*Resultat*'.

2.5.1.12. Números

Finalment, altra de les bases de paraules necessàries pel funcionament de la calculadora fa referència a la introducció dels diferents números que intervenen en el càlcul. Amb aquest propòsit, a la funció '`condicions.m`' s'ha definit un vector anomenat '*Numeros*' que es guardarà a l'arxiu '*condicions*' i que té com a components els noms de les diferents paraules necessàries per poder introduir els números a la calculadora. El contingut d'aquest vector és:

'*0*', '*1*', '*2*', '*3*', '*4*', '*5*', '*6*', '*7*', '*8*', '*9*', '*Negatiu*', '*Coma*', '*Composicio*'

2.5.2. Grava

Mitjançant aquesta funció és possible enregistrar un arxiu d'àudio de tipus '`.wav`' amb el nom que desitjem. La capçalera associada a aquesta funció és:

```
grava(nom,temps)
```

L'arxiu '`.wav`' es guardarà amb el nom especificat a al variable '`nom`'. L'enregistrament serà de tipus '*mono*' i de durada igual a la quantitat en segons assignada a la variable '`temps`'.

Per procedir a l'enregistrament sonor, es fa servir la funció de Matlab '`wavrecord`' la qual necessita com a paràmetres la freqüència de mostreig i el número de mostres que cal guardar.

Per a obtenir la freqüència de mostreig la funció '`grava`' fa ús de les condicions del sistema, per tant carrega l'arxiu '*condicions*'. Queda així assignada automàticament una freqüència de mostreig de 8000 mostres per segon. Amb el producte del temps de la variable '`temps`' i la freqüència de mostreig tindrem les mostres que calen guardar.

Hem comentat en l'apartat 2.4.1.5. que en fer un enregistrament sonor, es genera un pic que queda gravat al començament de l'arxiu '`.wav`'. S'estima que aquest pic ha desaparegut passades unes 2000 mostres. Per tal d'eliminar aquest pic, es farà l'enregistrament tot augmentant el nombre de mostres de la gravació amb aquestes 2000 mostres que després eliminarem.

Per generar el nou arxiu d'àudio sense el pic, copiem a un nou vector totes les mostres de l'enregistrament sense les 2000 primeres i exportem les mostres d'aquest vector a un arxiu '`.wav`' amb el nom de la variable '`nom`' tot fent servir la funció de Matlab '`wavwrite`.

Finalment la funció '`grava`' retorna el nom de l'arxiu d'àudio.

a Exemple

Per enregistrar un arxiu d'audio de 3 segons amb el nom '*paraula.wav*', podríem escriure a la finestra de comandes '*Command Window*':

```
>> grava('paraula',3)
```

2.5.3. tallaParaula

Quan es realitza un enregistrament d'una paraula amb la funció 'grava', es genera un arxiu '.wav. Aquest arxiu tindrà enregistrada la paraula pronunciada però també tindrà enregistrats espais de silenci. Depenent de la durada de l'enregistrament i de la llargària de la paraula hi haurà més o menys espais de silenci. D'altra banda és molt difícil que la distància del locutor al micròfon sigui sempre la mateixa. Per tant és molt probable que l'amplitud de les paraules enregistrades variï en successius enregistraments. Ens interessa tenir un arxiu on la paraula '*ompli*' tot l'arxiu sense silencis i que totes les paraules tinguin el mateix marge dinàmic. Amb aquest objectiu s'ha desenvolupat la funció tallaParaula.m. Com el seu nom indica aquesta funció pretén deixar la paraula '*tallada*' sense silencis i al mateix temps normalitzar-la.

Hem de tenir en compte l'objectiu d'aquesta funció és de deixar la paraula '*preparada*' (normalitzada i sense silencis) per iniciar després una segmentació de la mateixa en trames. Aquestes trames es formaran '*enfinestrant*' la paraula ja preparada. Les finestres tenen un ample fix i determinat pel sistema. Per tant alhora de '*tallar*' els silencis i deixar la paraula preparada, anirà bé que aquesta finalment tingui un nombre de mostres múltiple del nombre de mostres que componen una finestra. És a dir tallarem els silencis en '*unitats de finestra*'.

Per tant el primer que ens interessa saber és el nombre sencer de finestres que cabran a la paraula. Per aquest motiu cal saber el número de mostres que hi ha en una finestra. Això ho sabrem multiplicant l'amplada de finestra del sistema (en segons) per la freqüència de mostreig del sistema. Si recordem ambdues dades són condicions generals del sistema i com a tals estan guardades en memòria en l'arxiu '*condicions*' generat amb la funció 'condicions.m'. Per tant serà necessari carregar l'arxiu '*condicions*'.

Arribat aquest punt es pot procedir a normalitzar la paraula. Tot seguit comença el procés d'eliminació de silencis.

Per poder eliminar els silencis establirem uns llindars que ens delimitaran les '*zones de tall*'. Aquests llindars es trobaran a partir de mesures de potència. Necessitarem dues mesures de potència: *potència de soroll* i *potència del senyal*. Per fer mesurament de la potència del soroll hem d'establir una zona on sabem que no hi ha pronunciació de la paraula. Aquesta zona considerem que es situa en els primers 100 mil·lisegons. Aquest és un temps raonable de no pronunciació donat que es considera que el locutor encara no ha començat a parlar. Tot sabent el temps d'amplada de la finestra podrem saber el número de finestres que corresponen a la zona de soroll.

A continuació tot sabent el número sencer de finestres que hi cabran a la paraula i el número de mostres que conté cada finestra, es procedirà a calcular la potència mitja del senyal a cada finestra. Aquesta potència es calcularà segons l'equació de mitjana quadràtica (rms)[6, pàg. 22] [13, pàg. 2] donada la seva utilitat en variables que poden tenir tant números positius com negatius (donat que la mitjana podria obtenir valors nuls

o propers a zero):

$$P_f = \sqrt{\frac{\sum_{k=1}^{N} s(k)^2}{N}} \tag{2.2}$$

Es formarà un vector de components igual al número de finestres el qual conté la potència mitja de cada finestra. Per tant en aquest vector tindrem la potència mitja de la paraula distribuïda en finestres. A continuació podem calcular la mitjana de la potència mitja de les finestres tot fent mitjana dels valors d'aquest vector. Sabent quantes finestres ocupa la zona de soroll, calcularem la potència mitja en la zona de soroll tot fent la mitjana dels valors dels '*n*' primers components del vector igual a les '*n*' finestres de la zona de soroll.

Arribat aquest punt, podem fer un càlcul de la '*relació senyal soroll*' (SNR). Podem considerar la paraula enregistrada $s(k)$ com la suma d'un senyal '*missatge*' $x(k)$ amb un '*soroll aditiu independent*' $n(k)$. Tal que:

$$s(k) = x(k) + n(k) \tag{2.3}$$

Com que $x(k)$ i $n(k)$ són independents i com el soroll té valor mitjà zero, es demostra[7, pàg. 203]

$$P_s = P_x + P_n \tag{2.4}$$

entenent P_s com la potència mitja del senyal amb soroll de fons, P_x la potència mitja del senyal missatge útil i P_n la potència mitja de soroll.

El càlcul de relació senyal soroll (SNR) relaciona les potències del senyal missatge útil $x(k)$ i el soroll $n(k)$:

$$SNR = 10 \cdot \frac{P_x}{P_n} \tag{2.5}$$

Per tant, en el nostre cas al tractar-se de soroll aditiu:

$$SNR = 10 \cdot \log\left(\frac{max(P_s) - max(P_n)}{max(P_n)}\right) \tag{2.6}$$

on $max(P_s)$ és la potència màxima de la paraula i $max(P_n)$ és la potència màxima a la zona de soroll.

Aquest càlcul de SNR és d'utilitat. Si $SNR \geq 0$ equival a dir que la potència de la paraula amb soroll de fons P_s és el doble o més que la potència de la zona de soroll P_n o en termes del senyal, que *la potència del senyal és superior a la del soroll*. Aquest és el criteri que s'ha utilitzat per continuar amb el procés. Si la potència del senyal no supera la potència del soroll, s'ha considerat que la paraula no mereix ser tractada i la funció '`tallaParaula`' retornarà un 1. Aquesta discriminació servirà per indicar al sistema que la paraula necessita ser enregistrada una altra vegada. D'altra banda si es supera aquesta condició, s'ha considerat que l'enregistrament de la paraula és òptim i, per tant, es procedeix a trobar els '*punts de tall*' per eliminar els silencis.

Per establir els punts de tall, un cop tenim el càlcul de la potència mitja de totes les finestres i de les finestres de soroll, es buscarà la primera finestra que iguali o sobrepassi la potència mitja. Un cop trobada, aquesta finestra serà un primer punt de fixació 'P_1'. A continuació es tracta de trobar la darrera finestra anterior a P_1 que tingui una potència menor o igual a la potència mitja de soroll. Per tant des de P_1 anirem *retrocedint* fins

trobar aquesta finestra. Un cop trobada, aquesta finestra serà el primer punt de '*tall*' de la paraula T_1. Tot seguit es procedirà de forma anàloga però començant pel final de la paraula. Per tant, des del final i retrocedint, es buscarà primer la primera finestra que tingui una potència igual o superior a la potència mitja. Aquesta finestra ens proporcionarà el segon punt de fixació 'P_2'. En aquest moment és important remarcar que aquest retrocés no ha d'arribar fins a P_1. Tot seguit es tractarà de buscar *cap endavant* per trobar la primera finestra posterior a 'P_2' que tingui una potència menor o igual a la potència mitja de soroll. Un cop trobada aquesta finestra s'haurà trobat el segon '*punt de tall*' de la paraula T_2. Les mostres de la paraula '*tallada*' seran les corresponents a les mostres de les finestres entre T_1 i T_2.

2.5.4. grava_Base

L'entrenament el sistema és una fase prèvia i necessària pel reconeixement de paraules. Per tal d'entrenar el sistema es requereix d'un enregistrament de totes les possibles paraules a reconèixer. Totes les paraules del sistema '*ParLab*' estan agrupades formant diversos conjunts que hem anomenat '*bases*'. Per tan quan s'enregistri una paraula per la fase d'entrenament, serà d'utilitat disposar d'una funció que pugui decidir si una paraula és òptima per ser guardada i, donat el cas, deixar-la '*preparada*' i ubicada a la seva base corresponent. Alhora de desenvolupar el complement '*ParLab*' s'ha decidit que les paraules corresponents a una mateixa base es guardin a un mateix directori. Així doncs, cada base tindrà el seu propi directori.

La funció '*grava_Base*' guarda una paraula '*preparada*' a un directori assignat per a una base concreta. Escrivint a la pantalla de comandes:

```
>> grava_Base(nomDirectori,nomParaula);
```

procediríem a l'enregistrament d'una paraula de nom '*nomParaula*' que quedaria guardada i '*preparada*' al directori '*nomDirectori*.

Com que necessita enregistrar una paraula, necessita saber el temps de l'enregistrament. Recordem que el temps assignat per enregistrar una paraula del sistema el conté la variable '*tempsParaula*' declarada i assignada a la funció '`condicions.m`' i guardada a l'arxiu '*condicions*'. Per tant serà necessari carregar l'arxiu '*condicions*'. En cridar la funció '*grava_Base*' aquesta crida a la funció '*grava*' per realitzar un enregistrament[1] amb el nom de la paraula i directori. Tot seguit la funció '*grava_Base*' es disposa a fer un **control** sobre l'enregistrament realitzat per decidir si és òptim per a ser '*preparat*' o no. Aquest '*control*' el determina la condició de relació senyal soroll que fa la funció '*tallaParaula*'. Per tant, tot seguit es crida a la funció '*tallaParaula* i es comprova que aquesta no retorna un 1. Recordem que si la funció '*tallaParaula*' retorna un 1 significa que no supera la condició de relació senyal soroll ($SNR < 0$) i no ha preparat la paraula. Quan la funció '*grava_Base*' té aquesta informació, adverteix mitjançant un missatge sonor que hi ha error de qualitat del senyal i repeteix la operació de l'enregistrament fins que aconsegueixi tenir una paraula enregistrada en condicions '*òptimes*'.

[1] Recordem que la funció '*grava*' 'tallarà' el pic que es genera i per tant elimina les 2000 primeres mostres.

Quan ja disposa d'una paraula enregistrada en condicions '*òptimes*', significa que també la té '*preparada*' per la funció '*tallaParaula*'. Per tant ara la paraula està normalitzada, amb silencis '*tallats*' i amb una longitud proporcional a l'amplada de la finestra que intervindrà després per generar trames. Tot seguit la funció '*grava_Base*' pot guardar un arxiu '`.wav`' amb la paraula '*preparada*' al directori de la base.

2.5.5. grava_Mostra

Quan s'executa el complement '*ParLab*' mitjançant el botó '*Executar*' podem pronunciar una paraula corresponent a una funció o una tasca a realitzar. Per tant a partir d'aquest punt el sistema tindrà una paraula '*entrant*'. Aquesta paraula entrant és la que haurà de reconèixer. Un cop reconeguda, s'executarà la tasca corresponent, i la paraula pronunciada i enregistrada, ja no es necessitarà més. Per tant cada vegada que entri una paraula al sistema es considera que és un '*àudio de mostra*' per ser identificat. La propera paraula que entri al sistema al prémer el botó '*Executar*' també serà un '*àudio de mostra*' entrant substituint a l'anterior que es va pronunciar.

Per tant el sistema necessita una funció que enregistri una paraula entrant a la qual se li assignarà sempre el mateix nom ('*AudioMostra*'), que pugui decidir si aquesta paraula entrant és òptima per ser processada i en cas afirmatiu, deixar la paraula '*preparada*' per la posterior segmentació en trames.

De forma molt similar a la funció '*grava_Base*', la funció '*grava_Mostra*' utilitza les condicions del sistema de l'arxiu '*condicions*' per fer un enregistrament d'una paraula amb el nom '*AudioMostra*' tot fent servir la funció '*grava*'. De seguit, crida a la funció '*tallaParaula*' i fa un **control** de qualitat a través de la mesura de la relació senyal soroll que li proporciona aquesta funció. Anàlogament a la funció '*grava_Base*' si no supera la condició de relació senyal soroll, avisarà per mitjà d'un missatge d'àudio i caldrà tornar a repetir l'enregistrament fins que es superi aquesta condició. Un cop superada significa que la funció '*tallaParaula*' ha '*preparat*' la paraula i ja està normalitzada, sense silencis i amb una longitud proporcional a l'amplada de finestra que s'utilitzarà després per generar trames. Finalment, la funció '*grava_Mostra* guarda la paraula '*preparada*' en un arxiu d'àudio de tipus '`.wav`' al directori del complement '*ParLab*' i de nom '*AudioMostra*'.

2.5.6. emfasi

Aquesta és la funció prèvia a la segmentació en trames per l'extracció de característiques de la paraula. Un cop el sistema disposa d'una paraula enregistrada en condicions òptimes i '*preparada*', és a dir, normalitzada, sense silencis i amb una longitud múltiple de l'amplada de la finestra que s'utilitzarà per segmentar en trames, cal aplicar-li un tractament previ. Aquest és el filtre de preèmfasi.

Com hem vist a la part teòrica, en pronunciar una paraula, fins que arriba al micròfon el senyal pateix una atenuació d'uns 6 dB per octava [4, pàg. 40]. Per tant la paraula pateix més atenuació a les freqüències agudes i aquestes són importants donat que ens proporcionen informació sobre els formants. Per tal de compensar aquesta atenuació es

dissenya un filtre de primer ordre anomenat '*filtre de preèmfasi*'. Aquest es pot considerar com un tipus de filtre passa altes.

Tot fent servir la funció '*filter*' de Matlab, dissenyem el filtre de preèmfasi. El paràmetre del filtre a acostuma a tenir valors entre 0,95 i 1 [4, pàg. 40] . Quan més proper és el valor de a a 1, més gran és l'efecte de preèmfasi. Pel nostre cas, hem escollit un valor mig de **0,97**.

Un cop tenim l'equació de la transformada Z del filtre, per implementar-lo en Matlab, fem un vector amb els coeficients de la transformada Z, aquest vector serà:

```
v=[1 -0.97]
```

a continuació podem filtrar el nostre senyal `X` amb la instrucció filter:

```
filter(v,1,X)
```

La funció ens retornarà les mostres de la paraula filtrada.

La figura 2.5 ens mostra la resposta en freqüència del filtre que s'ha dissenyat. Per

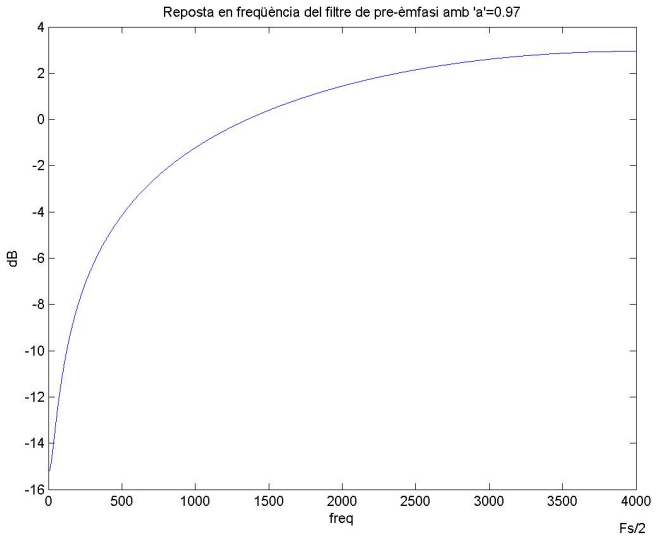

Figura 2.5: Resposta en freqüència del filtre de preèmfasi amb a=0,97

aconseguir-la podem fer servir la funció del Matlab `freqz` de forma semblant a com hem fet servir la funció `filter`, amb el vector `v` i el número de punts que desitgem visualitzar `N`, on el darrer punt `N` correspon amb la meitat de la freqüència de mostreig $\frac{F_s}{2}$ (en el nostre cas 4000):

```
freqz(v,1,4000)
```

com que volem visualitzar el mòdul:

```
modul=abs(freqz(v,1,4000))
```

volem el valor en decibels (dB):

```
moduldB=10*log10(abs(freqz(v,1,4000)))
```

podem definir un vector de les mostres implicades:

```
x=[1:1:4000];
```

En aquest vector el darrer punt correspon a la meitat de la freqüència de mostreig que és a la vegada la freqüència de tall en Hz per garantir el criteri de '*Nyquist*'.

finalment generar la gràfica amb la funció '*plot*':

```
plot(x,moduldB)
```

2.5.7. winCeps

Un cop tenim la paraula normalitzada, sense silencis i s'ha compensat l'atenucaió que pateixen les freqüències agudes mitjançant el filtre de preèmfasi, s'està en disposició de començar a extreure les característiques.

El primer pas consisteix en fer trames de la paraula. Per aconseguir aquestes trames, es segmentarà la paraula fent servir finestres tipus '*Hamming*'. Degut la característica atenuació que presenta en els extrems la finestra '*Hamming*', aquestes trames necessitaran un solapament. Pel nostre cas hem assignat una longitud de finestra de **20 mil·lisegons** i una distància entre finestres de **10 mil·lisegons**. Recordem que aquests paràmetres estan assignats a les variables '`AF`' i '`SepF`' respectivament que es troben a l'arxiu '*condicions*' juntament amb la freqüència de mostreig del sistema '`Fs`' de **8000** mostres per segon. Amb aquestes dades sabem que cal construir finestres de **160 mostres** d'amplada cada **80 mostres**.

S'ha programat una funció que va '*enfinestrant*' la paraula en finestres de tipus '*Hamming*'. Mitjançant la funció de Matlab '*window*' s'ha construit una finestra tipus '*Hamming*' com la que mostra la figura 2.6. Aquesta finestra s'utilitza per anar creant les diferents '*trames*' de la paraula. Un cop es té una finestra es calculen els '*coeficients Cepstrum*' de la finestra fent servir la funció del Matlab '*rceps*'. La funció '*rceps*' calcula tot d'una el **cepstrum real** de tota la finestra, això significa que d'un sol pas calcula la part real de l'antitransformada del logaritme del mòdul de la transformada de Fourier de la finestra, és a dir `real(ifft(log(abs(fft(finestra)))))`. Mitjançant la operació cepstrum, s'ha aconseguit la '*transformació homomòrfica*' que *ens permet tenir distribuïdes en temps les característiques de l'espectre de freqüència de la trama de la paraula*.

Observem pas per pas el que succeeix en la transformació '*cepstrum*'. Quan entra una paraula normalitzada i sense silencis i passada pel filtre de preèmfasi es segmenta en trames. Una trama equival a una finestra tipus '*Hamming*' de la paraula. La figura 2.7 mostra la trama número 8 de la paraula '*calculadora*' de 20 mil·lisegons on es pot apreciar l'atenuació del senyal als extrems de la trama degut a l'enfinestrat tipus '*Hamming*'. També es pot apreciar en aquest cas concret un patró periòdic d'aproximadament 7,5 mil·lisegons.

La figura 2.8 representa el mòdul de la transformada de Fourier de la mateixa trama, on

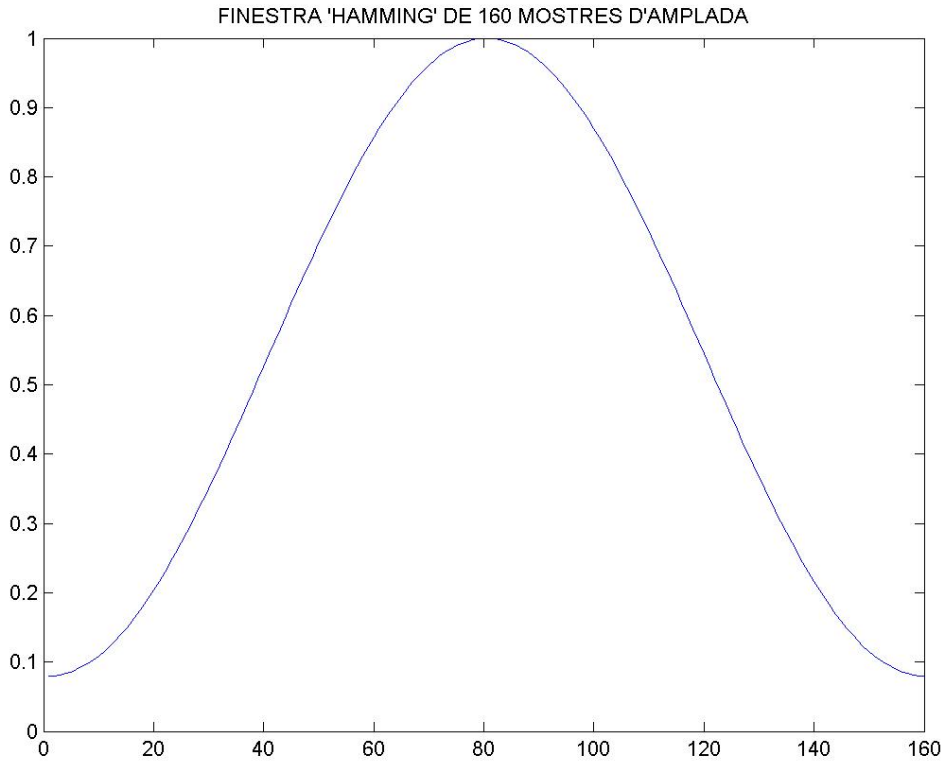

Figura 2.6: Finestra '*Hamming*' de 160 mostres

podem veure les diferències respecte al marge dinàmic de les diferents freqüències. Notem que alguns formants poseeixen alts components de freqüència i que distingir els diferents formants és prioritari per distingir diferents paraules. Aquestes diferències respecte al marge dinàmic queden reduïdes al aplicar la funció logaritme.

Tal com mostra la figura 2.9 al aplicar logaritme al mòdul de la transformada de Fourier, es redueix el marge dinàmic i també les diferències entre les components de freqüències agudes i greus. En aquest punt també es pot observar que l'espectre té una forma semblant a una ona '*modulada en amplitud*'. La 'moduladora' ens aporta informació sobre la manera en la qual es dóna forma al tracte vocal per construir els formants. D'altra banda, la 'portadora' (ja que es pot considerar quasi periòdica) ens aporta informació sobre el període fonamental del senyal de veu (to de veu), característica més pròpia del locutor. Destaca la gràfica en color vermell del la figura 2.9 que segueix la 'moduladora'.

Tot seguit, s'aplica l'antitransformada de Fourier sobre el logaritme i es retorna al domini del temps tal com mostra la figura 2.10. Es proposa observar la figura 2.10 com si es tractés d'un espectre, és a dir com si el temps fos freqüència. Per no confondre'ns amb la freqüència 'vertadera' i sabent que estem en realitat en el domini del temps, a aquesta hipotètica 'freqüència' li direm '*qüefrència*[2]. Podem veure com els components de baixes '*qüefrències*' corresponen les freqüències de la 'moduladora' i que a '*qüefrències* més

[2]Anàlogament a la consecució del terme 'cepstrum' tot seguint el joc d'inversió de les primeres lletres de 'spectrum', s'inverteixen les dues primeres síl·labes de la paraula 'freqüència' per denominar qüefrència al seu domini

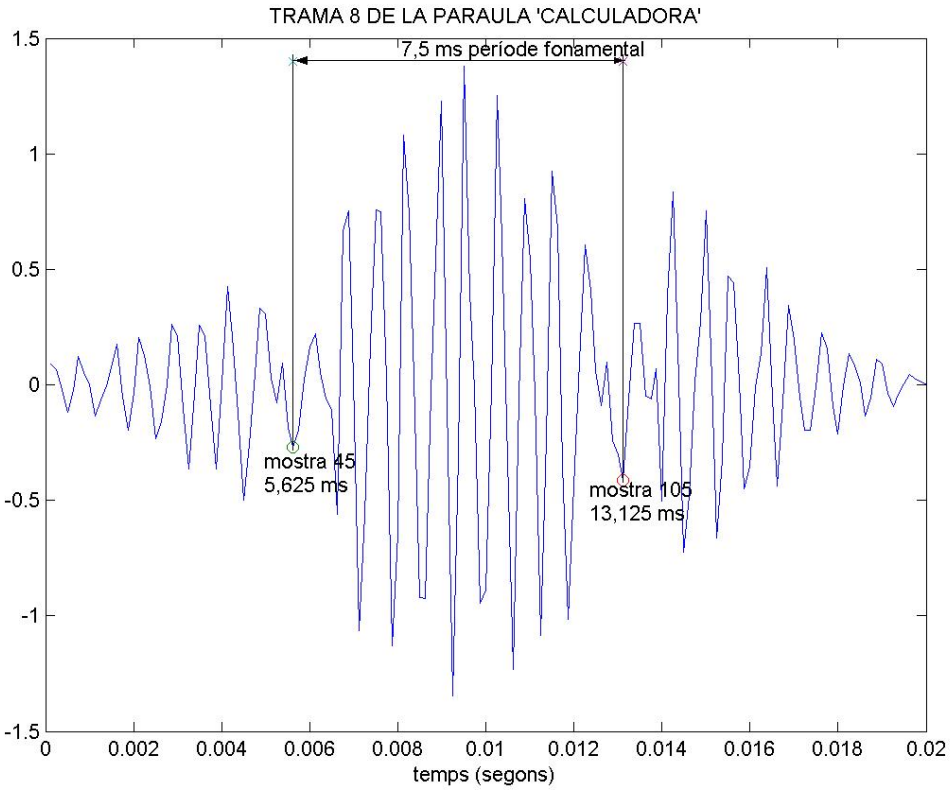

Figura 2.7: Trama 8 de la paraula '*Calculadora*'. Període fonamental d'un patró periòdic dins la trama.

altes tenim el '*rahmònic*'[3] corresponent al to fonamental de la 'portadora'. Per tant la part real de l'antitransformada ha separat les components del tracte vocal i del to de veu, és a dir les característiques pròpies dels formants de la paraula i les característiques pròpies del locutor, completant-se així la '*transformació homomòrfica*'.

Un cop tenim el cepstrum, es poden seleccionar mitjançant un '*filtrat en temps*', també anomenat '*liftat*'[3, pàg. 126] les components que més es desitgin[4]. Si volem identificar paraules, ens quedarem amb les components de baixes '*qüefrències*', si volguéssim identificar locutors ens quedaríem amb components d'altes '*qüefrències*'. En el nostre cas ens interessa identificar paraules i per tant ens interessa la informació de la 'moduladora' del tracte vocal, per aquest motiu filtrarem i ens quedarem amb els **10 primers coeficients Cepstrum**. Recordem que la variable que conté el número de coeficients cepstrum que desitgem està continguda a l'arxiu '*condicions*' amb el nom 'NC'. Aquests 10 coeficients ens haurien de proporcionar informació de la 'moduladora' del tracte vocal, fet que es pot comprovar a la figura 2.9 en la gràfica de color vermell. La *gràfica de color vermell és la gràfica dels 10 primers coeficients cepstrum de la trama 8 de la paraula 'calculadora'*. Estirada al llarg de la freqüència i superposada a la gràfica de color blau corresponent al logaritme del mòdul de la transformada, podem comprovar com efectivament aquests 10 coeficients s'ajusten a la 'moduladora' d'aquest senyal, de fet és com si haguéssim filtrat

[3] Anàlogament al terme 'qüefrència' anomenarem '*rahmònics*' als 'harmònics' del cepstrum

[4] El terme '*liftat*' prové de la inversió de les primeres lletres del terme anglès'*filtering*' quan es realitza per separar components cepstrals.[3]

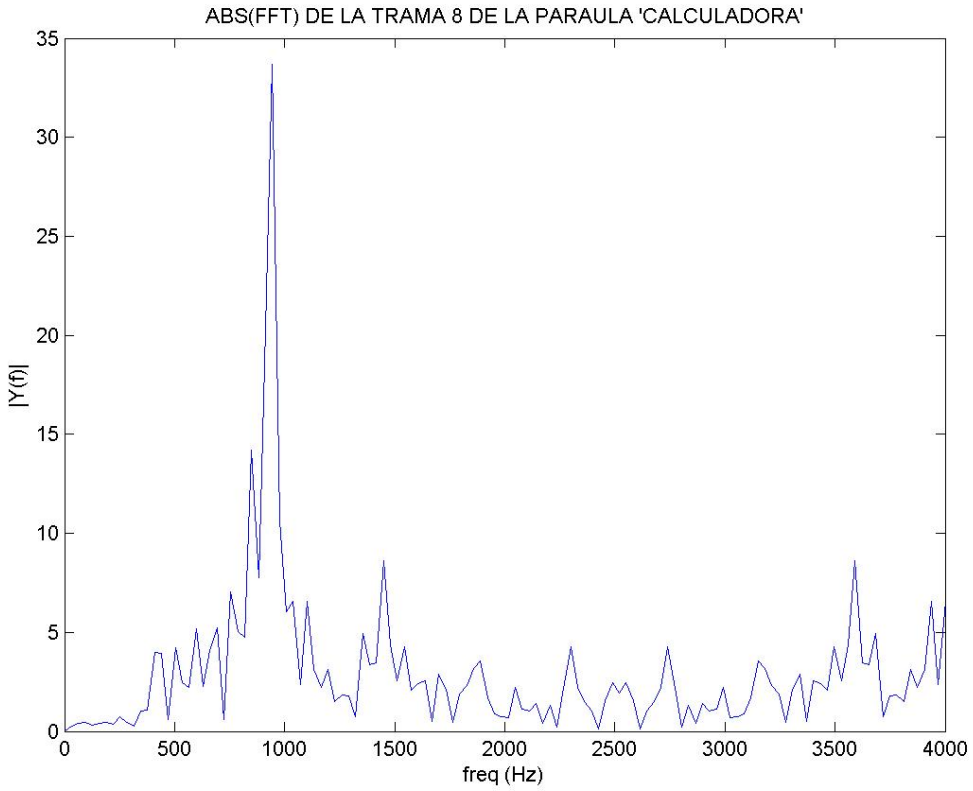

Figura 2.8: Mòdul de la transformada de Fourier de la trama 8 de la paraula '*Calculadora*'.

aquest senyal per quedar-nos amb la 'moduladora'.

Aquests 10 coeficients són els que anirem guardant en una matriu, on cada fila d'aquesta matriu contindrà els 10 primers coeficients d'una trama de la paraula. Per tant la matriu tindrà tantes files com trames tingui la paraula. Finalment la funció `winCeps` retorna la matriu que conté els 10 coeficients cepstrum de totes les trames de la paraula.

2.5.8. deltaCeps

Un cop tenim els coeficients cepstrum, tenim les diferents característiques de cada trama. Cada trama té un espectre propi del qual s'han extret uns components de modulació del tracte vocal que aporten informació sobre la formació dels fonemes. Fins aquí tenim informació de les diferents trames per separat, constituint els que s'acostuma anomenar com '*coeficients estàtics*'.

Els diferents fonemes que formen una paraula aporten informació sobre ella, però també aporta informació *la variabilitat que tenen aquests fonemes en el temps*. Alhora de pronunciar, no és el mateix allargar la durada d'uns fonemes més que altres dintre d'una mateixa paraula, que pronunciar-los tots a la mateixa durada. Aquestes característiques són també importants alhora de reconèixer una paraula. Fent un estudi de la variació que ténen aquestes trames en funció del temps originen uns altres coeficients que s'acostu-

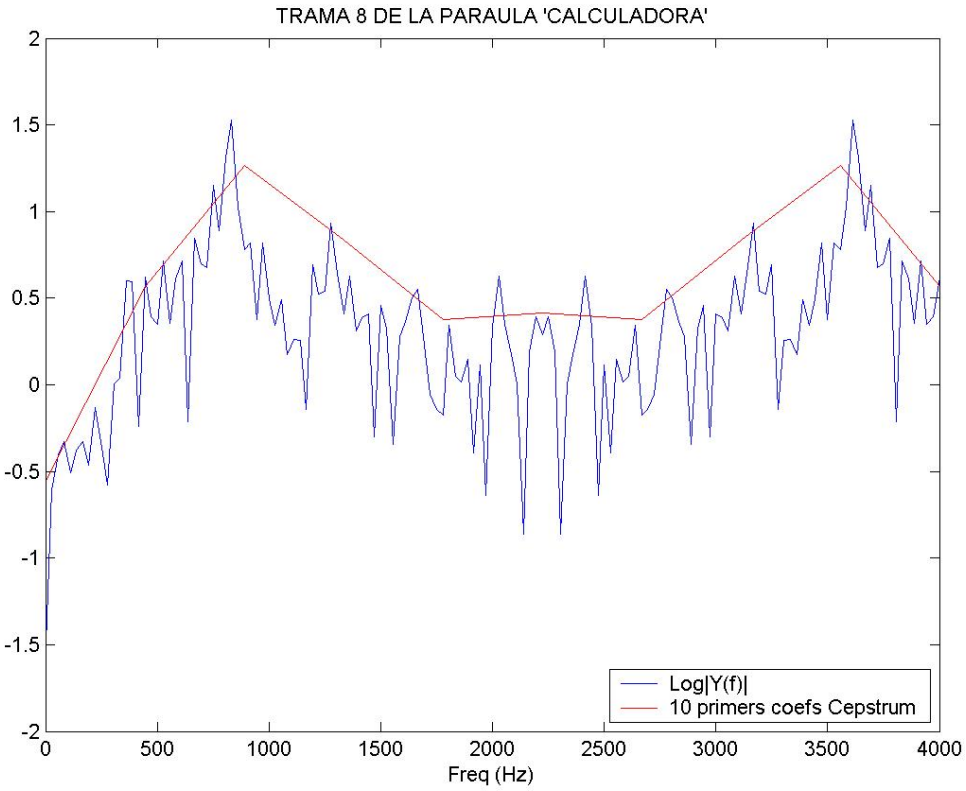

Figura 2.9: Logaritme del mòdul de la transformada de Fourier de la trama 8 de la paraula '*Calculadora*' en blau. El gràfic vermell representa els 10 primers coeficients cesptrum de la trama.

men a anomenar '*coeficients dinàmics*' o també '*delta-cepstrum*'.

La funció '`deltaCeps`' fa regressió lineal sobre la matriu que conté els diferents coeficients cesptrum de totes les trames per anar obtenint els diferents coeficients '*delta-cepstrum*' per cada trama. Aquesta regressió lineal és realitzada en un interval d'estimació determinat pel paràmetre '`ND`' guardat a l'arxiu '*condicions* (en el nostre cas 2, originant un interval de 2ND+1=5 trames). Per cada trama n farà una regressió lineal tenint en compte les dues trames anteriors i les dues posteriors. Un cop té el càlcul de la regressió lineal per una trama, amplia la matriu de característiques afegint als components cepstrum els components delta-cepstrum per cada trama. Aquesta nova matriu de components cepstrum i delta-cepstrum de totes les trames serà la matriu final de les '*característiques*' de la paraula.

2.5.9. procesaBase

Durant la fase d'entrenament, un primer pas és enregistrar les paraules d'una base. Aquesta tasca hem vist que la realitza la funció '*grava_Base*'. El segon pas és processar aquestes paraules per extreure les característiques de cada una d'elles. La funció '`procesaBase.m`' processa totes les paraules del directori d'una base i guarda les carac-

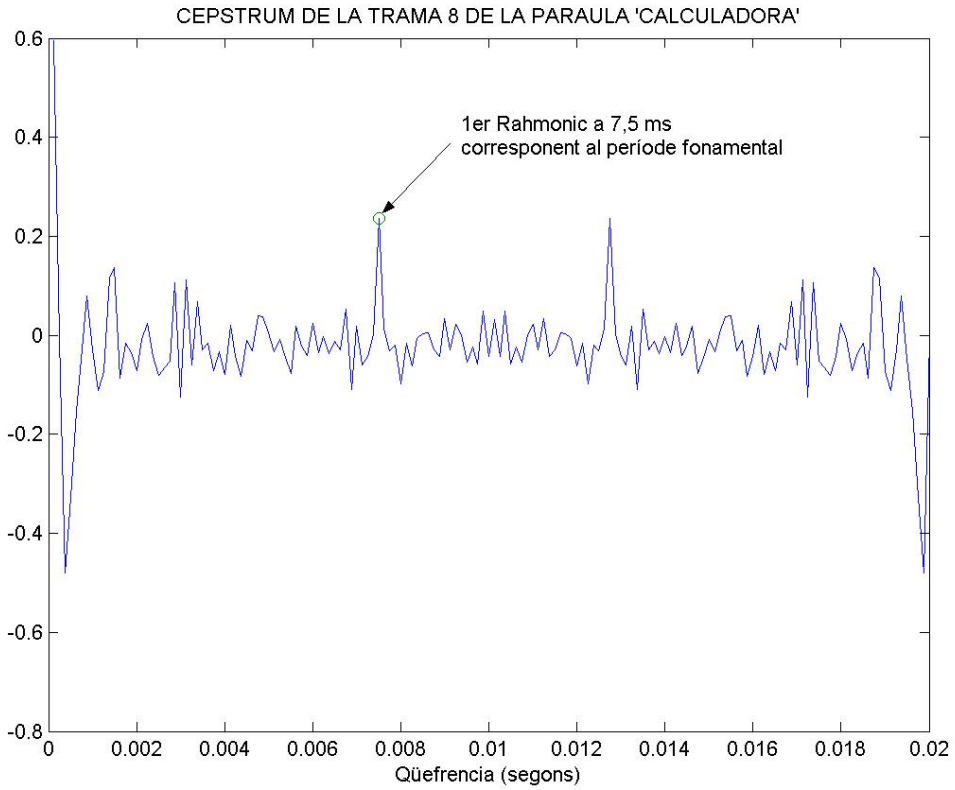

Figura 2.10: Cepstrum de la trama 8 de la paraula '*Calculadora*'. Observem el primer '*rahmonic*' situat a 7,5 ms i corresponent al to de veu de període fonamental 7,5 ms

terístiques de cada una d'elles en un altre directori assignat per a la base ja processada.

El nom del directori on es troben les paraules i del directori on quedaran processades s'especifica a la capçalera de la funció de la forma: `procesaBase(nomDirBase,nomDirProc)` On '`nomDirBase`' és el nom del directori de la base a processar i '`nombDirProc`' és el nom del directori on es guardaran les paraules processades de la mateixa base. El sistema està pensat per que cada base tingui el seu propi directori destí de base 'processada'.

El mecanisme de la funció es entrar dintre del directori de la base, i anar processant cada paraula una a una fins processar totes les paraules del directori. Per cada paraula, carrega les mostres amb la funció de Maltlab '*wavread*'. Recordem que cada paraula ve '*acondicionada*' sense silencis i normalitzada (perquè ja està enregistrada a la base). A continuació s'aplica el filtre de '*preèmfasi*' amb la funció '`emfasi.m`', es divideix en trames i s'extreuen els 10 primers coeficients '*cepstrum*' de cada trama mitjançant la funció '`winCeps.m`', i finalment s'acaba d'extreure les característiques ampliant amb la obtenció dels coeficients '*delta-cepstrum*' mitjançant la funció '`deltaCeps.m`'.

Finalment la funció guarda per cada paraula un arxiu que conté la matriu de les característiques generada per la funció `delta-cepstrum.m` en la variable '`dc`' i les mostres de la paraula sense extreure les característiques en la variable '`ptall`'.

2.5.10. DTW

Alhora de cercar una paraula, el sistema necessitarà comparar les característiques d'una paraula entrant amb les característiques de les paraules contingudes a una base per tal d'escollir aquella que més se sembli. La paraula entrant es processa per extreure les característiques. Tindrem doncs una matriu de totes les trames de la paraula i les seves característiques. A continuació per cercar la paraula més semblant cal comparar les seves característiques amb les característiques de cada una de les paraules de la base. A l'hora de comparar dues paraules, estarem comparant dues matrius, i el número de files de cada matriu correspon al número de trames de cada paraula. El problema és que el número de files de cada matriu a comparar és diferent donat que les paraules a comparar són de diferent llargària i per tant tenen diferent número de trames. Fins i tot en el cas de paraules que siguin iguals, és gairebé impossible que s'hagin enregistrat a la mateixa velocitat.

Per tant necessitem una manera de comparar dues matrius de diferent número de files.

Tal com hem vist a la part teòrica, el '*Dynamic Time Warping*' permet '*alinear*' dues seqüències dependents del temps i que per fer-ho utilitza mesures de distància. També s'ha vist que amb aquesta mesura de distància es pot generar una matriu de '*costos*' acumulats i que quan més se semblen dues seqüències, menor és el cost acumulat a aquesta matriu.

La funció '*DTW' calcula la matriu de costos acumulats del 'Dynamic Time Warping'* entre dues matrius de característiques de dues paraules diferents i ens retorna el darrer cost acumulat. La funció '`DTW.m`' amb capçalera,

```
DTW(A,B)
```

calcula el cost tot comparant cada fila (que conté les característiques d'una trama) de la matriu `A` amb totes les files de la matriu `B`. Per cada comparació valora un cost aplicant la '*distància euclídea*' al quadrat. Com que ens interessa la valoració de costos per comparar, fem servir la distància euclídea al quadrat donat que simplifica còmputs i en aquest cas ens fa la mateixa funció que la distància euclídea[3, pàg. 132]. Un cop té calculat el cost, busca a la matriu de costos '`D`' el mínim cost anterior a distància d'una fila, o columna, o d'una fila i una columna i l'acumula al cost calculat per anar omplint la matriu de costos acumulats. Com hem vist a la part teòrica, finalment el cost '*total* acumulat està al darrer component de la matriu de costos. Aquest és el valor que ens interessa obtenir. Per tant la funció retornarà el valor del **darrer cost acumulat**. En la mesura que més se sembli la paraula, menor serà aquest valor.

2.5.11. entraMatriu

Quan s'ha executat el complement '*ParLab*' i s'ha pronunciat la paraula '*calculadora*', i aquesta és reconeguda pel sistema, entrem en el mode calculadora i el sistema necessita de nous paràmetres. Com hem vist a la secció 2.4.1.7. la calculadora no sap quants

números ni de quin tipus[5] volem inserir. Per tant necessitem una funció que ens permeti anar enregistrant els diferents paràmetres i que sàpiga quan ja no volem enregistrar més. Aquesta és la tasca de la funció '`entraMatriu.m`'.

De forma semblant a la funció '*grava_Base*', la funció '*entraMatriu*' anirà fent diversos enregistraments successius corresponents a diferents paraules. La diferència és que en principi no sap quantes paraules enregistrarem. Per tant la funció '*entraMatriu*' comença amb un so d'un to pur per avisar al locutor del començament de l'enregistrament de la primera paraula. Es procedeix a l'enregistrament fent servir la funció '*grava*'. A continuació es normalitza i s'eliminen els silencis mitjançant la funció '*tallaParaula*'. Recordem que la funció '*tallaParaula* estableix un criteri amb un control de la relació senyal soroll per decidir si la paraula '*mereix*' ser guardada o no i que en cas negatiu, la funció retornarà un 1. Aquest és precisament el control que executa la funció '*entraMatriu*' per continuar enregistrant paraules. Donat que mentre es compleixi el criteri de relació senyal soroll, es procedirà al següent enregistrament. Si no volem continuar enregistrant, simplement haurem de guardar silenci, i al no complir amb el criteri la funció entendrà que no desitgem enregistrar més paraules. En aquest punt la funció '*entraMatriu*' emetrà un so d'un to més agut per indicar que ha finalitzat la sèrie d'enregistraments i esborrarà el darrer enregistrament fet, donat que correspon a silenci. Com hem vist a la secció 2.4.1.7., els paràmetres de la calculadora cal que siguin dictats en un ordre determinat. Aquest ordre és el que permetrà després fer una cerca ordenada de cada un dels paràmetres. Per tal efecte, quan la funció '*entraMatriu*' inicia la sèrie d'enregistraments, aquests guardaran al directori que hem anomenat '*numeros*' cada arxiu '`.wav`' amb el nom de l'ordre en què s'ha dictat. És a dir el primer paràmetre de la calculadora que es dicti, s'anomenarà '`1.wav`', el segon '`2.wav`' i així successivament.

2.5.12. cercaCalc

Un cop hem entrat a la calculadora tot pronunciant la paraula '*calculadora*' després de prémer el botó '*executar*' del complement '*ParLab*' i hem 'dictat' els paràmetres un a un fent ús de la funció '*entraMatriu*' per a realitzar un càlcul, el sistema ha de reconèixer cada una de les paraules que hem enregistrat per obtenir els paràmetres de la calculadora. Aquest reconeixement és el que durà a terme la funció '*cercaCalc*'.

Com hem vist a la secció 2.4.1.7., els paràmetres de la calculadora cal que siguin 'dictats' en un ordre específic. També hem vist que la funció '*entraMatriu*' guarda els enregistraments corresponents als diferents paràmetres en ordre tot assignant el número de l'ordre en què s'ha pronunciat al nom de l'arxiu '`.wav`'.

La funció '*cercaCalc*' anirà obtenint del directori '*numeros*' cada paraula en l'ordre en que van ser dictades. Per tant, primer carregarà les mostres de l'arxiu anomenat '`1.wav`' fent servir la funció '*wavread*'. Al saber l'ordre de la paraula, pot saber a quin directori desenvolupar el reconeixement de la paraula. Per tant si es tracta de la paraula '`1.wav`', donat que el primer paràmetre correspon a *l'operació a realitzar*, realitzarà el reconeixement al directori o base on es tenen processades les diferents paraules que configuren les ope-

[5] hem vist a la secció 2.4.1.7. que per inserir números de vàries xifres, negatius o bé decimals es necessiten inserir diferents paràmetres.

racions, és a dir a la base ‘operacions’. Per tant cada paraula enregistrada per la funció ‘*entraMatriu*’ al directori ‘*numeros*’ es considera com una paraula o ‘*àudio mostra*’ entrant al sistema per ser reconegut en una base concreta.

Un cop sap a quin directori ha de fer el reconeixement, inicia la fase de reconeixement de la paraula seguint les etapes següents:

1. Ha d’extreure les característiques de la paraula entrant.
   (a) Aplica el filtre de preèmfasi fent servir la funció ‘*emfasi*’.
   (b) Segmenta en trames i obté els 10 primers coeficients cepstrum de cada trama, fent servir la funció ‘*winCeps*’.
   (c) Finalitza la obtenció de les característiques ampliant amb l’obtenció dels coeficients delta-cepstrum, fent servir la funció ‘*deltaCeps*’.
2. Un cop té les característiques inicia la comparació amb cada una de les paraules de la base corresponent:
   (a) Calcula la matriu de costos acumulats del ’Dynamic Time Warping’ entre dues paraules utilitzant la funció ‘*DTW*’ i guarda el valor del cost acumulat a un vector. Repeteix la operació tot comparant amb totes les paraules de la base per ordre i guardant cada cost acumulat al vector per ordre.
   (b) Selecciona el mínim del vector que conté els costos acumulats i obté la posició que ocupa aquest mínim. La posició que ocupa li diu quina paraula és la que té el cost més petit i per tant la que se sembla més. La posició marca l’ordre que ha anat agafant les paraules de la base per comparar-les. Per tant amb la posició obté el nom de la paraula que ocupa aquella posició dins la base.

Un cop ha reconegut els diferents paràmetres tradueix el nom de cada paraula trobada a un número i el passa a un vector. Aquest vector tindrà una sèrie de números corresponents a les diferents paraules trobades que configuren els paràmetres de la calculadora.

Per exemple si la operació a realitzar per la calculadora ha reconegut que es tracta de la paraula ‘*suma*’ a la primera component del vector assignarà el número 1. Si es tracta de ‘*resta*’ el 2, ‘*multiplica*’ el 3, ‘*divideix*’ el 4, ‘*potència* el 5. La segona component del vector correspon la segona paraula pronunciada i si és reconeguda com ‘*números*’ li assigna 1, si és ‘*resultat*’ 2. A partir de la tercera component aniran els diferents paràmetres que serviran per configurar els números. Per tant a partir de la tercera component assignarà el valor del número identificat de la base ‘*números*’ i si ha reconegut la paraula ‘*negatiu*’ assignarà el número 10, si la paraula és ‘*coma*’ assignarà el 11 i si es tracta de la paraula ‘*composició*’ assignarà un 12.

Finalment la funció ‘*cercaCalc*’ retorna el vector amb tots els números que codifiquen els paràmetres de la calculadora.

2.5.13. vector

Un cop la funció '*cercaCalc*' ha reconegut els diferents paràmetres de la calculadora i els ha codificat en un vector, necessitem una funció que tradueixi la codificació dels paràmetres en números. Recordem que en el vector que retorna la funció '*cercaCalc*' hi ha paràmetres que serveixen per generar els números, paràmetres com ara el número 10 que significa que el número serà negatiu o bé el número 12 que significa que el número serà una composició de vàries xifres. Per tant es necessària una conversió d'aquests paràmetres que configuren els números als números pròpiament dits. Aquesta és la tasca que fa la funció '*vector*'. La funció vector '*reconstrueix*' els números que li proporciona el vector retornat per la funció '*cercaCalc*' i *retorna un altre vector amb els números reconstruïts*.

2.5.14. llegirNum

Aquesta funció dicta xifra a xifra un número. Es tracta d'una contestació per part del sistema cap al locutor. Per tant la màquina utilitza la mateixa codificació que ha fet servir el locutor a l'hora de dictar números. La funció, a partir d'un número qualsevol fa una codificació i genera un vector amb el número codificat segons els paràmetres que hauria d'utilitzar un locutor per inserir el número a la calculadora. Tot seguit la funció reprodueix l'arxiu '`.wav`' associat a cada número sense reproduir les paraules '*composició* ni el número de xifres que composa el número ni tampoc el número de xifres decimals. La funció retorna el vector del número codificat.

Per exemple si volem que ens dicti el número 134.765 podem escriure a la finestra de comandes '*Command Window*':

```
>>llegirNum(134.765)
```

obtindrem com a resposta el vector:

```
12      3      1      3      4     11      3      7      6      5
```

i tot seguit ens reproduirà les paraules: '*un*', '*tres*', '*quatre*', '*coma*', '*set*', '*sis*', '*cinc*'.

2.5.15. calcula

Un cop la calculadora té tots els paràmetres amb els números reconstruïts com el vector que proporciona la funció '*vector*', es poden iniciar els càlculs. A partir del vector que retorna la funció '*vector*', de la primera component obté la operació a realitzar, de la segona component obté si necessita incorporar a la operació el darrer resultat o no, i de la resta de components obté els números que intervenen en el càlcul totalment reconstruïts. Amb tots aquests paràmetres realitza el càlcul. Tot seguit dicta el resultat del càlcul mitjançant la funció '*llegirNum*'. Finalment la funció guarda en un arxiu el resultat del càlcul per poder-lo incorporar en una posterior execució de la '*calculadora*'. La funció retorna un vector amb

la codificació corresponent al tipus d'operació realitzada, la opció d'incorporar resultat o no, els números que han intervingut a la operació i el resultat.

2.5.16. ensenya_resultat

Un cop una paraula entrant al sistema és reconeguda com una de les sis principals tasques de la base '*funcions*' citades a la secció 2.3.6. és necessari que s'executin les tasques pertinents.

La funció '*ensenya_resultat* a partir del nom de la paraula trobada, reproduirà un arxiu d'àudio tipus '`.wav`' associat a la paraula reconeguda per informar del seu reconeixement. Tot seguit executarà les comandes corresponents i assignarà a una variable el nom de la paraula reconeguda. Per exemple en el cas de la paraula '*neteja*' executarà la comanda '*home*' i la funció retornarà la cadena de caràcters '`neteja`'.

Si la paraula reconeguda és '*calculadora*', la funció '*ensenya_resultat*' cridarà a la funció '*entraMatriu*' per tal d'iniciar els enregistraments dels diferents paràmetres necessaris per la calculadora. Tot seguit mitjançant la funció '*cercaCalc*' obtindrà el vector de números que codifiquen tots els paràmetres de la calculadora. Un cop té els paràmetres de la calculadora, cridarà la funció '*vector*' per descodificar el vector proporcionat i així tenir en un nou vector els paràmetres i els números que intervindran en el càlcul. A continuació pot realitzar el càlcul cridant la funció '*calcula*', mitjançant la qual se'ns dictarà el resultat i obtindrem en un vector la informació corresponent a la operació realitzada, si ha utilitzat el resultat anterior o no, els números que han intervingut a la operació i el resultat. Amb aquesta informació, construeix un vector de caràcters on assignarà el símbol de la operació realitzada, les lletres '*R*' o '*N*' en funció de si s'ha fet servir el darrer resultat o no, i els números i resultat convertits a format de caràcter. Finalment la funció '*ensenya_resultat* retornarà aquest vector on cada component és cada una d'aquestes cadenes de caràcters. Aquesta informació serà utilitzada per mostrar-la en les pantalles de la interfície gràfica del component '*ParLab*'.

2.5.17. CercaParaula

Un cop premem el botó '*executar*' i enregistrem una paraula mitjançant la funció '*grava_Mostra*', entra una paraula al sistema per ser reconeguda. La funció que posa en marxa tota la tasca de reconèixer la paraula i executar les diferents comandes que li pertoquen a cada paraula reconeguda és '*CercaParaula*'. La funció '*CercaParaula*' inclou totes les tasques que corresponen als blocs d'extracció de característiques i comparació amb altres paraules de la base a partir d'una paraula entrant. El diagrama de la figura 2.11 ens mostra un resum de les diferents funcions del sistema que intervenen fins aquest procés.

El primer pas per reconèixer una paraula entrant és l'extracció de les seves característiques.

1. Ha d'extreure les característiques de la paraula entrant.

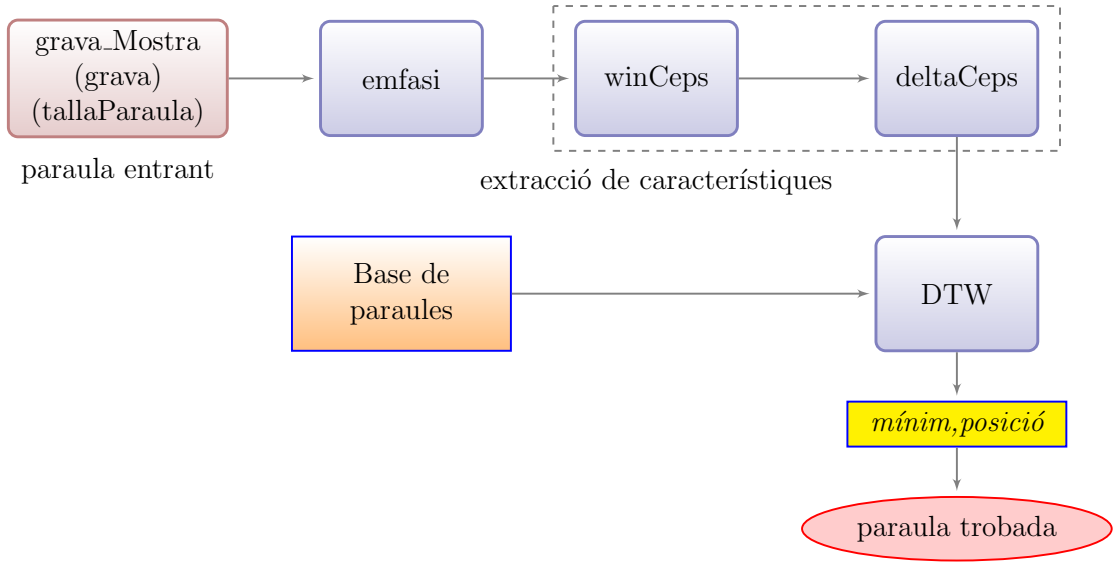

Figura 2.11: Diferents funcions del sistema '*ParLab*' que intervenen en la fase de reconeixement

(a) Aplica el filtre de preèmfasi fent servir la funció '*emfasi*'.

(b) Segmenta en trames i obté els 10 primers coeficients cepstrum de cada trama, fent servir la funció '*winCeps*'.

(c) Finalitza la obtenció de les característiques ampliant amb l'obtenció dels coeficients delta-cepstrum, fent servir la funció '*deltaCeps*'.

2. Un cop té les característiques inicia la comparació amb cada una de les paraules del directori '*WAV_Proc*' que és on estan totes les paraules de les funcions principals del sistema processades.

   (a) Calcula la matriu de costos acumulats del 'Dynamic Time Warping' entre dues paraules utilitzant la funció '*DTW*' i guarda el valor del cost acumulat a un vector. Repeteix la operació tot comparant amb totes les paraules de la base per ordre i guardant cada cost acumulat al vector per ordre.

3. Selecciona el mínim del vector que conté els costos acumulats i obté la posició que ocupa aquest mínim. La posició que ocupa li diu quina paraula és la que té el cost més petit i per tant la que se sembla més. La posició marca l'ordre que ha anat agafant les paraules de la base per comparar-les. Per tant amb la posició obté el nom de la paraula que ocupa aquella posició dins la base.

4. Executa les comandes associades a la paraula trobada i rep informació de la paraula per mostrar-la a pantalla tot fent servir la funció '*ensenya_resultat*'.

Finalment la funció retorna el vector de caràcters amb components de paraules que li retorna la funció '*ensenya_resultat* per mostrar aquesta informació posteriorment a pantalla.

2.5.18. Executar

Cada vegada que es vulgui executar alguna comanda mitjançant la veu caldrà prémer el botó '*executar*' de la interfície gràfica del complement '*ParLab*'. Quan es posa en funcionament l'execució del sistema, es necessita una tasca gestioni automàticament tot el procés. Aquesta és la tasca de la funció '*executar*'.

Fent servir les condicions dels sistema que conté l'arxiu '*condicions*' emet un so d'un to pur de mig segon de durada per avisar de l'inici de l'enregistrament de la paraula entrant. Tot seguit crida a la funció '*grava_Mostra*' per aconseguir enregistrar una paraula entrant al sistema. Recordem que la funció '*grava_Mostra*' farà un control de qualitat de l'enregistrament i si no és òptim obligarà a repetir l'enregistrament. Finalment el sistema disposarà d'una paraula entrant que compleix els criteris de qualitat i que està normalitzada i sense silencis.

A continuació es fa el reconeixement de la paraula fent servir la funció '*CercaParaula*'. Recordem que la funció '*CercaParaula*' trobarà la paraula que més se sembli a la paraula entrant, executarà les comandes associades a ella i retornarà un vector amb informació per mostrar a pantalla.

Aquest vector retornat per la funció '*CercaParaula*' és el que finalment retornarà la funció '*executar*'.

2.5.19. Notes

La funció '*notes*' assigna una nota musical a un número del 0 al 12. Per tant hi ha 13 notes musicals seleccionades de l'escala de Do Major en l'interval que va del '*sol 2*' fins al '*sol 4*'. La primera nota associada al número 0 és el '*sol 2*' i la segona associada al número 1 és el '*do 3*', generant-se així entre el 0 i el 1 un interval de quarta ascendent per simular la cadència conclusiva corresponent als graus de dominant-tònica de la tonalitat de Do Major. A partir de la nota associada al número 1 (do 3) fins la associada al número 12 (sol 4) estan ordenades tot seguint l'escala natural fins assolir la octava i la dominant del registre superior. D'aquesta manera, en repetir la seqüència de notes arribant a la darrera (sol 4) i tornant a començar per la nota 0 (sol 3) i a continuació la nota 1 (do 3) es conclou amb la octava de la dominant (sol 4) per anar a l'octava baixa de la dominant (sol 2) per concloure amb la tònica (do 3) produint-se una transició agradable de cadència conclusiva per tornar a començar de nou. D'altra banda la cadència conclusiva ens ajudarà a identificar quan comença de nou la sèrie.

Aquests sons de les diferents notes corresponen als diferents sons que s'escoltaran al prémer els botons de '*par +*', '*par -*', '*base +*' o '*base -*' en la interfície gràfica de '*ParLab*', per poder configurar les diferents bases o modificar les diferents paraules d'una base, donat que cada base i cada paraula d'una base té associada una nota musical. Depenent del número de bases o de paraules d'una base hi haurà més o menys registre de notes que podran sonar. Si per exemple la base de '*operacions*' consta de cinc paraules, quan seleccionem les diferents paraules a modificar podran sonar fins a cinc notes (del do 3 al sol 3 una associada a cada paraula).

2.6. ParLab

La funció de la interfície gràfica del complement '*ParLab*' es diu 'ParLab.m'. Així doncs per activar la interfície gràfica escriurem a la finestra de comandes '*Command Window*'

```
>> parlab
```

La funció '*ParLab*' porta associades a cada comandament o botó de la interfície gràfica diferents funcions de les anteriorment descrites per assolir les diferents tasques de cada comandament definides a la secció 2.4.1..

2.6.0.1. Configurar

Primerament el botó configurar fent servir el nom de la base a configurar que consta a la pantalla de '*base*' comprova si és un nom de base vàlid, és a dir si hi ha paraules associades a aquesta base. Si no és correcte, emetrà un missatge sonor advertint de l'existència de l'error. Si és correcte se li assignarà el directori on són les paraules a processar de la base i el directori on haurà de guardar les paraules processades de la mateixa base. A continuació fent servir les condicions del sistema de l'arxiu '*condicions*' obtindrà informació sobre el número de paraules de la base i quins noms tenen. Tot seguit emetrà un missatge sonor d'un arxiu '.wav' amb les instruccions a seguir i es disposarà a fer la sèrie d'enregistraments de les diferents paraules de la base per mitjà de la funció '*grava_Base*'. Per cada paraula farà sonar un so d'un to pur de mig segon de durada per avisar del començament de l'enregistrament i cridarà a la funció '*grava_Base*' per enregistrar-la amb un criteri de qualitat, normalitzada i sense silencis.

Quan hagi enregistrat totes les paraules, emetrà un altre so corresponent a un to pur però més agut que el d'inici i cridarà a la funció '*procesaBase*' per extreure totes les característiques de totes les paraules de la base i guardar-les a un mateix directori assignat per a aquesta base.

Finalment per mitjçà de la funció del Matlab 'wavplay' emetrà el missatge d'àudio 'configurat.wav' per avisar que el procés ha acabat.

2.6.0.2. Modificar

Per modificar una paraula d'una base, la funció '*Modificar*' assigna la informació corresponent al nom de la paraula a modificar i de la base a la qual pertany de les caixes de text (o "*pantalles*") corresponents a '*paraula*' i'*base*' respectivament de la interfície gràfica. Un cop té aquests noms se li assigna els directoris de la base on enregistrarà la paraula i el directori on es guarden les paraules processades de la mateixa base. Tot seguit fa una comprovació per verificar que la paraula seleccionada a la caixa de text paraula pertany a la base de la caixa de text base. Si no és correcte emet el missatge sonor d'error 'Error_Dades.wav' per mitjà de la funció de Matlab '*wavplay*'. Si és correcte proceceix a fer l'enregistrament '*òptim*' de la paraula cridant a la funció '*grava_Base*'. Genera un so d'un to pur per indicar que comença d'enregistrament de la paraula, tot seguit fa servir la

funció '*grava_Base*' per realitzar l'enregistrament amb un criteri de qualitat, normalitzada i sense silencis. Un cop ha enregistrat la paraula emet un genera i emet un so d'un to pur més agut per indicar que ha finalitzat l'enregistrament. Seguidament crida a la funció '*procesaBase*' per processar altra vegada totes les paraules de la base, extreure les característiques de cada paraula i guardar-les al directori on són les paraules processades de la mateixa base.

Finalment per mitjà de la funció del Matlab '`wavplay`' emetrà el missatge d'àudio '`Modificat.wav`' per avisar que el procés ha acabat.

2.6.0.3. Executar

Aquesta funció posa en funcionament tot el sistema de reconeixement. La funció a desenvolupar en la interfície és senzilla donat que la tasca d'executar la realitza la funció '*executar*'. Comença amb un control del directori de les bases per comprovar que el sistema està entrenat. Tot seguit crida a la funció '*executar*'. Finalment mostra la informació corresponent (nom de paraula trobada) a les 'pantalles' de número i base.

2.6.0.4. Botons Par +, Base +

Aquestes funcions són molt semblants. Fan un recorregut per vectors que contenen els noms de les paraules o de les bases per mostrar-les a pantalla. Cada vegada que es prem un botó, es crida a la funció '*notes*' la qual emet un so corresponent a una nota musical.

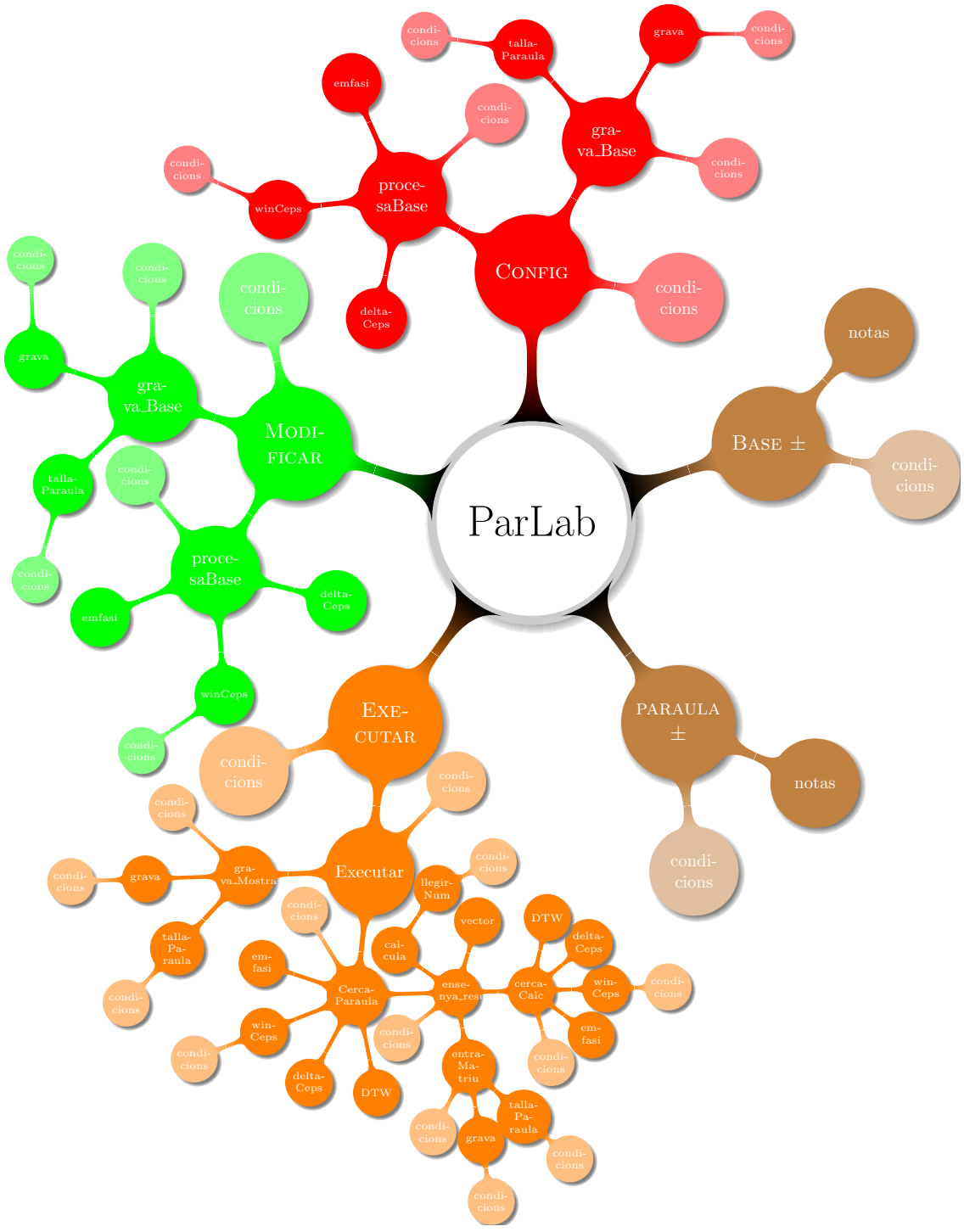

Figura 2.12: Diagrama de les diferents funcions associades als botons de '*ParLab*'

CAPÍTOL 3. RESULTATS I VALORACIÓ

Després de diverses proves, el sistema de reconeixement de paraules basat en cepstrum i desenvolupat utilitzant l'eina de càlcul Matlab donant lloc al complement '*ParLab*', presenta els següents resultats:

1. En el reconeixement de les sis tasques principals a realitzar per '*ParLab*', el sistema obté resultats satisfactoris en un 75% de les vegades.
2. Un cop el sistema reconeix la tasca '*calculadora*', entra en una segona i més complexa fase de reconeixement. Aquesta nova fase de reconeixement presenta els següents efectes:
   (a) Increment de la temperatura de l'ordinador.
   (b) Més dificultat a l'hora d'enregistrar els diferents paràmetres de la calculadora, donat que augmenta el soroll de fons reduint així la relació senyal soroll.
   (c) Quan més paràmetres requereix la '*calculadora*', més dificultat per poder introduir-los per mitjà de la parla.
   (d) Un cop introduïts tots els paràmetres de la calculadora, el sistema presenta més dificultat a l'hora de fer un reconeixement de totes les paraules de forma òptima. S'estima que aquest resultat està influenciat en gran mesura per:
      i. Al accedir a la '*calculadora*' per mitjà d'una fase prèvia de reconeixement de paraules conformant així una segona fase, s'incrementa considerablement la càrrega computacional del programa. Aquest fet repercuteix en la qualitat del senyal de les posteriors paraules enregistrades.
      ii. Intervenen més bases de paraules i una de les bases ('*números*') incrementa el nombre de paraules a reconèixer pel sistema a 13.
      iii. La base de major número de paraules conté les paraules de més curta durada (corresponents als noms de les xifres numèriques). La durada de les respectives paraules d'aquesta base és molt semblant i amb patrons fonètics similars (p.ex. "dos" i "tres", "cinc" i "sis") proporcionant poca diferenciació entre elles.
      iv. La pronunciació de les paraules corresponents als números a inserir a la calculadora es produeix al final de la seqüència de paràmetres quan l'ordinador més augmenta de temperatura i redueix les prestacions.
      v. En paraules de major número de fonemes i formant part de conjunts de paraules a comparar (bases) més petites, no es presenten tants problemes. Aquest és el cas de paraules com "*números*" o "*resultat*" de la base '*memòria*' o paraules relatives al tipus d'operació a realitzar. També paraules com "*composició*", "*coma*" o "*negatiu*" tot i formar part de la base de dades de major número de paraules, presenten més fonemes i llargàries diferents constituint així trets més distintius.
3. El sistema presenta una fàcil i còmoda interfície, proporcionant molta utilitat i flexibilitat a l'hora 'd'entrenar' el sistema.

4. La interfície ocupa poc espai en pantalla i requereix de molt pocs recursos per fer-la funcionar complint així amb un els requisits del disseny.

CAPÍTOL 4. CONCLUSIONS

1. Mitjançant aquest treball s'ha pogut comprovar que el so d'una paraula comporta diferents característiques, entre elles el to i els formants.

2. En la producció dels fonemes intervenen components a altes freqüències. Per aquest motiu és important emfatitzar-les si volem reconèixer millor les paraules.

3. El cepstrum és un tipus de transformació homomòrfica que permet separar aquests components de la parla i que podem seleccionar aquells que més interessi per tal de poder reconèixer paraules o persones.

4. Els sistemes de reconeixement de la parla basats en cepstrum tenen una càrrega computacional relativament petita i proporcionen un reconeixement acceptable en sistemes de poques paraules, no així en sistemes on el número de paraules a reconèixer sigui molt gran.

5. El Dynamic Time Warping 'DTW' és una tècnica que permet l'alineació de seqüències dependents del temps. Fet que permet la comparació de seqüències de diferent longitud.

6. Tot i proporcionar un reconeixement acceptable, hi ha vegades en que el sistema falla. Per tant no és aconsellable el reconeixement de la parla en el desenvolupament de sistemes que necessitin molta precisió en la elecció dels seus comandaments.

7. Matlab és una potent eina de càlcul que permet programar funcions amb un entorn que augmenta la possibilitat d'acció d'aquestes funcions. Aquest fet proporciona un increment de possibilitats de relació amb el mateix programari. Aquest era el propòsit del nostre treball.

8. Els sistemes de reconeixement de la parla proporcionen una manera diferent d'interactuar amb les màquines, resultant de gran utilitat en aplicacions diverses.

9. Aquesta manera diferent d'interactuar amb les màquines augmenta les possibilitats del seu comandament proporcionant eines de major utilitat. Aquest augment d'utilitat també proporciona un augment en la possibilitat de relació entre persones, millorant així la seva comunicació.

4.1. Futur

Hem vist que l'anàlisi cepstrum presenta un reconeixement acceptable en un nombre de paraules reduït. Un estudi de la manera de com els éssers humans perceben els sons ha conduït al desenvolupament d'altres tècniques que proporcionen major fiabilitat. Aquest és el cas dels *Melcepstrum Cepstral Coeficients*. Els coeficients basats en Melcepstrum parteixen de l'anàlisi cepstral per sotmetre a un banc de filtres simulant el comportament de l'oïda humana. Aquest procediment no ha estat objecte del present treball, no obstant es deixa indicació per a tenir en compte en possibles estudis posteriors.

Altres tècniques com els *Models ocults de Markov* o xarxes neuronals, amb una càrrega computacional major són orientades a sistemes amb vocabularis on intervenen major número de paraules.

CAPÍTOL 5. INFUÈNCIA DEL RECONEIXEMENT DE LA PARLA EN EL MEDI AMBIENT

En el desenvolupament d'aquest treball, s'ha pogut comprovar que és molt important disposar d'una paraula enregistrada amb una bona relació senyal soroll per tal que el sistema pugui dur a terme un reconeixement òptim. També s'ha pogut veure que l'espectre presenta components a altes freqüències que cal emfatitzar perquè ens aporten informació rellevant.

El soroll de fons '*contamina*' el senyal fent malbé precisament aquells components d'altes freqüències que ténen poca potència. De poc serveix emfatitzar-les si ja venen contaminades de soroll. El resultat és una distinció pobre i per tant un reconeixement erroni de la paraula. En aquest punt es fa notar la importància de rebre una paraula en '*bones condicions*' pel correcte funcionament del sistema. Aquest fet pot resultar un punt en contra dels sistemes basats en reconeixement de la parla, donat que posa de manifest la '*fragilitat*' d'un sistema que proporciona una interacció amb una màquina, que d'altra banda, s'espera que faci tasques de manera '*robusta*'.

Aquest fet, no obstant, no només és propi de les màquines. A les persones també ens passa. Quan una paraula la percebem amb una baixa qualitat el nostre cervell no aconsegueix diferenciar-la bé i, per tant, no entenem.

Al llarg de la història l'ésser humà ha fet servir l'enginy per desenvolupar eines que facin tasques que ell no és capaç d'assolir per sí sol. Però al mateix temps la manera en la que les fa servir ha establert un vincle de relació amb aquestes màquines. Aquesta relació el porta a percebre la màquina com quelcom diferent a un simple objecte.

L'evolució de l'ésser humà es basa en l'evolució de les seves creacions. Però la seva evolució creativa el porta a crear objectes cada cop més semblants a sí mateix. El límit esdevindria crear un '*alter ego*' omnipotent. En aquest límit, la interacció entre home i màquina seria igual que entre home i home. Per tant en el camí de l'adquisició d'aquest límit es situen els sistemes de reconeixement de la parla, de moment amb més limitacions de les que hi puguin haver entre persones, però en evolució per poder superar-les algun dia.

Ara tornem al present. És moment de pensar en millorar els sistemes actuals. Una manera de millorar-los és també prendre consciència de *tenir cura d'allò que els afecta*. És important desenvolupar sistemes robusts a entorns sorollosos, però *també és important reduir de soroll aquests ambients*. En aquest punt, les limitacions dels sistemes de reconeixement de la parla actuals ens posen en contacte amb els problemes que produeix un excessiu soroll. I és que el soroll fa malbé el sistema de reconeixement de paraules de les màquines i també de les persones. Aquest deteriorament del reconeixement afecta a la comunicació i per tant a la relació no tan sols entre persona i màquina sinó també entre les persones i tot el seu entorn, afectant així la seva qualitat de vida.

Segons el Professor Raes[14], "el soroll es pot definir com *un so no desitjat o un so molest i intempestiu que pot produir efectes fisiològics y psicològics, no desitjats en una persona*

o en un grup". Per tant el soroll és un so i la qualificació com a tal la produeix una valoració subjectiva del propi individu.

Les contaminacions acústiques provoquen danys físics i mentals. Entre els danys físics podem citar l'acceleració del pols, taquicàrdies, augment de la pressió arterial i mal de cap, augment del colesterol, problemes coronaris. També està comprovat que sorolls forts i sobtats poden causar fins i tot un infart. Entre els danys psicològics un dels principals és l'insomni, el qual genera fatiga i aquesta produeix la falta de concentració que porta a la poca productivitat.

Fins i tot hi ha ones que l'ésser humà no és capaç d'escoltar (infrasons) però que afecten al seu organisme podent arribar a causar la mort.

"Un informe de la Organització Mundial de la Salut (OMS) situa els 65 dB[14] com a límit superior desitjable" i un informe de la Unió Europea del 2005[16] diu que "80 milions de persones estan exposades diàriament a nivells de soroll ambiental superiors a 65 dB i altres 170 milions ho estan a nivells entre 55-65 dB". Altres estimacions[14] situen entorn als 113 milions la xifra de persones de la Unió Europea exposades per sobre de la tolerància límit de 65 dB.

Entre les causes que motiven el soroll hi ha la manca d'una correcta planificació en l'ús del sòl degut a un plantejament urbanístic erroni, un mal disseny el la traça de les vies de trànsit, mal aïllament acústic dels edificis residencials o bé dels que desenvolupen activitats d'alt nivell sonor (bars, discoteques. . .)o poca previsió a l'hora d'instal·lar segons quines fonts acústiques.

Les fonts acústiques que generen soroll es poden distingir entre aquelles que produeixen nivells molt alts que causen danys físics i aquelles que amb nivells més baixos afecten la salut i la vida diària de les persones. Entre els causants de danys físics estarien els sorolls d'origen industrial. Com a representants del segon grup podríem citar el trànsit urbà.

El soroll l'ha portat l'ésser humà, ell és el responsable i per tant ell és qui ha de posar solució.

Els problemes associats al soroll han anat augmentant en el transcurs dels anys amb la creixent industrialització. Donat aquest augment, els països desenvolupats han promulgat normes per combatre la contaminació acústica en ciutats.

Per afrontar aquesta problemàtica, alguns sectors han tingut consciència de reduir el soroll, per exemple tenint cura en el disseny de maquinària menys sorollosa. Altrament ha crescut una indústria paral·lela per combatre els efectes del soroll, com producció de proteccions per les oïdes o la producció de materials aïllants. D'altra banda hi ha territoris que han promogut iniciatives per conscienciar a les persones per tal de reduir el nivell de soroll que elles mateixes generen. Darrerament hi ha sectors socials que han acumulat crítiques al seu comportament considerant-lo com a font de generació de soroll.

Donades totes aquestes circumstàncies es podria entendre que els sistemes de reconeixement de la parla poden ser més vulnerables en la societat actual. Precisament aquesta '*fragilitat*' li atorga un punt a favor de la conscienciació social per la reducció de soroll. Estenent l'ús dels sistemes de reconeixement de la parla actuals a moltes aplicacions

d'àmbits molt diferents, faria que molts sectors de la població fossin conscients de la importància de la reducció del soroll ambient. Per tant podem concloure que les limitacions dels sistemes de reconeixement de la parla actuals inclouen el punt a favor de contribuir de manera indirecta al medi ambient.

BIBLIOGRAFIA

[1] Farrús i Cabeceran, M. *Fusing prosodic and acoustic information for speaker recognition*. (PhD Dissertation, Hernando Pericás, F.J., Universitat Politècnica de Catalunya, Departament of Signal Theory and Communications, Barcelona, July 2008)

[2] Furui, S. *Speaker independent isolated word recognition using dynamic features of speech spectrum*. *IEEE Transactions on Acoustics, Speech and Signal Processing*, (Vol. ASSP-34, No.1, February 1986): 52–59.

[3] Hernando Pericás, F.J. "Medidas de distancia robustas". *Técnicas de procesado y representación de la señal de voz para el reconocimiento del habla en ambientes ruidosos*. (Tesis Doctoral, Nadeu i Camprubi, C., Universitat Politècnica de Catalunya, Departament de Teoria del Senyal y Comunicacions, Barcelona, mayo 1993): 123–148 http://www.tdx.cat/handle/10803/6911

[4] Young, S., Bloothooft, G. *Corpus-Based Methods in Language and Speech Processing* (Kluwer Academic Publishers. Dordrecht. 1997)

[5] Proakis, J. G., Manolakis, D. G. "Análisis frecuencial de señales y sistemas". *Tratamiento digital de señales, 3a ed.* (Prentice Hall. Madrid. 1998): 235–399

[6] Haykin, S., Van Veen, B. *Señales y sistemas.* (Limusa Wiley. México, D.F. 2003)

[7] Herrera Pérez, E. "Señales aleatorias y ruido". *Comunicaciones II, Comunicación digital y ruido: Una introducción a la teoría de la comunicación digital y el ruido.* (Limusa, Noriega Editores. México, D.F. 2004): 171–223

[8] Nuñez Batalla, F., Suárez Nieto, C. *Espectografía clínica de la voz* (Universidad de Oviedo. Servicio de Publicaciones. 1999)

[9] Valencia García, G. Orduña Bustamante, F. "El papel del sonido en nuestra noción del tiempo y el espacio". *Tiempo y espacio: miradas múltiples*. (Universidad Nacional Autónoma de México UNAM. Plaza y Valdés editores. México, DF. 2005.): 61–89

[10] Ouakrim, O. "Capítulo II, Consideraciones Teóricas" *Fonética y Fonología del Bereber*. (Universitat Autònoma de Barcelona, Servei de Publicacions. Bellaterra. 1995): 21–34

[11] Monroy-Casas, R. "Sobre la duración vocálica en español". *Aspectos fonéticos de las vocales españolas.* (LibrosEnRed. Amertown International S.A. 2005): 18–54

[12] Oppenheim, A. V., Schafer, R. W. "From Frequency to Quefrency: A History of the Cepstrum". *IEEE Signal Processing Magazine*. **1053-5888**(04), 95-99. (2004)

[13] Martin Hrnčár, M.Sc. "Voice command control for mobile robots". (University of Zilina, Faculty of Electrical Engineering. Slovakia) http://dsp.vscht.cz/konference_matlab/MATLAB08/prispevky/048_hrncar.pdf: 1–7

[14] De Esteban Alonso, A. "Contaminación acústica y salud". (Universidad Rey Juan Carlos, Facultad de Sociología, Instituto Universitario de Ciencias Ambientales (UCM)) http://revistas.ucm.es/index.php/OBMD/article/view/OBMD0303110073A/21658

[15] Muller, M. *Information Retrieval for Music and Motion.* (Ch. 4) (Springer. 2007):69–84 **ISBN 978-3-540-74047-6**. http://www.springer.com/cda/content/document/cda_downloaddocument/9783540740476-1.pdf?SGWID=0-0-45-452103-p173751818

[16] Wikipedia https://es.wikipedia.org

[17] “Diccionario de la Lengua Española” (Espasa-Calpe, 2005) http://www.wordreference.com

[18] http://www.logopediapsicologia.com

[19] Fotografia de laringe: “Laringe 1 por Mauricio A Mora” http://lmoralesco.blogspot.com.es/2011/05/membranas-ligamentos-y-articulaciones.html

[20] Fotografia del òrgans que intervenen en la fonació: http://roble.pntic.mec.es/msanto1/lengua/1sofolet.htm

APÈNDIXS

TÍTOL DEL TFC : Programa per al control de Matlab mitjançant la parla

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Sistemes de Telecomunicació

AUTOR: Luis Vázquez Alejandre

DIRECTOR: Gabriel Montoro López

DATA: 22 de setembre de 2015

APÈNDIX A. CODI DE PROGRAMA

A.1. condicions.m

```
Fs=8000; %Frequencia de mostreig
%Fs=11025;
tempsParaula=3; %temps de gravacio de una paraula
tempsResultat=15; %temps de gravacio del resultat
AF= 20e-3; %Amplada de la finestra en segons
SepF= 10e-3; %Separacio entre finestres en segons
NC=10; %Numero de coefecients Cepstrum
ND=2; %Index de coeficients Cepstrum a tenir en compte en el calcul
      %d' un coeficient Delta-Cepstrum. El numero de coeficients
  %sera 2*ND+1
nPS=6; %numero de paraules del sistema
%%%%%%%%%%%%%%
%BASES I PARAULES DEL SISTEMA
%%%%%
Bases={'Funcions','Numeros','Operacions','Memoria'};
Funcions={'Calculadora','Neteja','Workspace','Directori','Preferencies','Ajuda'};
Numeros={'0','1','2','3','4','5','6','7','8','9','Negatiu','Coma','Composicio'};
Operacions={'Suma','Resta','Multiplica','Divideix','Potencia'};
Memoria={'Numeros','Resultat'};
%%%%%
save('condicions','Fs','tempsParaula','tempsResultat','AF','SepF',
'NC','ND','nPS','Bases','Funcions','Numeros','Operacions','Memoria');
```

A.2. grava

```
%grava(nom,temps)
%Grava un arxiu d'audio de tipus .wav amb el nom especificat per la
%variable "nom".
%La grabacio es mono de tants segons com indica la variable 'temps'
%i a la frequencia de 8000 mostres per segon.
%P.ex. Per gravar un arxiu de 3 segons de nom "audio.wav" al
%directori on es troba la funcio grava.m teclejarem:
%       >> grava('audio',3);
%Per gravar un audio a una carpeta diferent, haurem d'especificar
%el directori de la manera:
%       >> grava('NomDirectori\audio',3);
%La funcio retorna el nom de la gravacio.
```

```
%nom='WAV_Base\AudioMostra';
function paraula = grava(nom,temps)
%Fs=8000;
load condicions;
talla=2000; %nombre de mostres que eliminarem per tallar el pic
r=wavrecord(talla+temps*Fs,Fs); %gravem tants segons com indiqui
%la variable 'temps' + les mostres de 'talla' que després
%eliminarem
%eliminem les 'talla' primeres mostres per tallar el pic
r2=r(talla+1:1:end);
wavwrite(r2,Fs,nom);
paraula=nom;
```

A.3. tallaParaula

```
function ptallada = tallaParaula(nom)
load condicions;
paraula=wavread(nom);
N=AF*Fs; %Numero de mostres per finestra
finestres=floor(numel(paraula)/N); %numero de finestres (sense guarda)
Tn=100e-3; %temps de mesurament de condicions de soroll (els primers 100 ms).
nfn=Tn/AF; %numero de finestres de mesurament de condicions de sorroll.
zcn=0; %creuaments acumulats en la zona de mesura de soroll
en=0; %energia acumulada en la zona de mesura de soroll
es=0;
pnor=paraula./max(abs(paraula)); %normalitzem
%Numero de Creuaments per Zero 'CZ' per finestra
for i=0:finestres-1 %index de finestra
    CZ=0;
    Ep=0;
    for k=1:N-1
        CZ=CZ+abs(sign(pnor(N*i+k+1)-pnor(N*i+k))); %creuaments per zero
        Ep=Ep+pnor(N*i+k)^2; %energia parcial
    end
    creuaments(i+1)=CZ; %vector de creuaments per zero de cada finestra
    energia(i+1)=sqrt(Ep/N); %vector de energia de cada finestra

end

zcnMig=sum(creuaments(1:1:nfn))/nfn; %Mitja de creuaments per zero
%a la zona de mesura de soroll
enMig=sum(energia(1:1:nfn))/nfn; %Mitja de l'energia acumulada a la
%zona de mesura de soroll
esMig=sum(energia)/finestres; %Mitja de l'energia acumulada de tota la senyal
%FIXEM ELS LLINDARS
ZCT=zcnMig; %fixem el llindar de creuaments per zero
```

```
ITL=enMig; %fixem el llindar inferior a 1.5 del valor de l'energia de soroll
ITU=esMig; %fixem el llindar superior a 1/2 de l'energia mitja
SNR=10*log((max(abs(energia))-max(abs(energia(1:1:nfn))))/max(abs(
energia(1:1:nfn))));
if SNR>=0 %si la potencia mitja del senyal es mes gran que la de soroll
%BUSQUEM ELS PUNTS (les finestres) QUE SOBREPASSEN ELS LLINDARS
j=1;
while (j<finestres)&&(energia(j)<ITU)
    j=j+1;
end
if (j<finestres) %si no arriba al final del vector es perque s'ha
%trobat el llindar ITU
    f1=j;
    while (f1>1)&&(energia(f1)>ITL) %Trobem f1: finestra de valor
%ITL anterior desde ITU
        f1=f1-1;
    end
    %while (j<finestres)&&(energia(j)>ITU) %busquem el proper valor
%que cau per sota ITU
    %    j=j+1;
    %end
    f=finestres;
    while(f>j)&&(energia(f)<ITU)
        f=f-1;
    end
    f2=f;
    while (f2<finestres)&&(energia(f2)>ITL) %Trobem f2: la finestra
%de valor ITL posterior a la caiguda per sota de ITU
        f2=f2+1;
    end
end

n1=f1*N+1; %mostres fins la finestra f1
n2=(f2-1)*N; %mostres fins la finestra f2
ptallada=pnor(n1:n2); %copiem la paraula entre els talls trobats
else
    ptallada=1;
end
```

A.4. grava_Base

```
%grava_Base(titol)
%Grava un arxiu d'audio de 3 segons amb el nom contingut a la variable "titol"
%a la carpeta WAV_Base.
%L'arxiu es mono de tipus wav i de frequencia de mostreig de 8000 mostres
```

```
%per segon tal com s'especifica a "grava.M".
%P.ex. si volem gravar un audio de nom "Elefant" a la base de dades
%escriurem:
%       >> x='Elefant';
%       >> grava_Base(x);
%O be asignant el titol de manera directa:
%        >> grava_Base('Elefant');
%Mes informacio teclejar
%>> Help grava

%nom='WAV_Base\AudioBase';
%grava(nom);
function audioGB = grava_Base(nomD,nomGB)
load condicions;
%nomD='WAV_Base\';
nomDGB=[nomD '\' nomGB]; %juntem els noms de l'arxiu i el directori
temps=tempsParaula; %gravarem 3 segons
%
n=[1:Fs/2]; %mostres de duracio del senyal d'inici 'sINI' (mig segon)
sINI=sin(2*440*n); %senyal sonora inici de cada paraula
%
audioGB=grava(nomDGB,temps); %gravem el arxiu amb el nom al directori
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SNR=tallaParaula(nomDGB); %tallem paraula i comprovem si SNR>=0
while SNR==1 %si SNR<0 la paraula no es talla i tallaparaula retorna 1
    missatge2=wavread('ErrorSNR.wav');
    wavplay(missatge2,Fs);
    wavplay(sINI,Fs);
    audioGB=grava(nomDGB,temps);
    SNR=tallaParaula(nomDGB);
end
wavwrite(SNR,Fs,nomDGB); %guardem la paraula tallada amb SNR>=0
```

A.5. grava_Mostra

```
%Grava un arxiu d'audio de 3 segons amb el nom "AudioMostra.wav" a
%la carpeta del directori on es troba "Grava_Mostra.M".
%L'arxiu es mono de tipus wav i de frequencia de mostreig de 8000 mostres
%per segon tal com s'especifica a "grava.M".
%Mes informacio teclejar
%>> Help grava

nom='AudioMostra';
load condicions;
```

```
temps=tempsParaula; %gravarem 3 segons
n=[1:Fs/2]; %mostres de duracio del senyal d'inici 'sINI' (mig segon)
sINI=sin(2*440*n); %senyal sonora inici de cada paraula
grava(nom,temps);
SNR=tallaParaula(nom); %tallem paraula i comprovem si SNR>=0
while SNR==1 %si SNR<0 la paraula no es talla i tallaparaula retorna 1
    missatge2=wavread('ErrorSNR.wav');
    wavplay(missatge2,Fs);
    wavplay(sINI,Fs);
    grava(nom,temps);
    SNR=tallaParaula(nom);
end
wavwrite(SNR,Fs,nom); %guardem la paraula tallada amb SNR>=0
```

A.6. emfasi

```
%Funcio Emfasi
%Aplica un filtre de pre-emfasi al senyal d'entrada X amb parametre 0.97
%Aixi doncs si volem que Y sigui un senyal resultant d'aplicar un filtre de
%pre-emfasi al senyal d'entrada X amb parametre 0.97, escriurem:
%Y=emfasi(X)
%
function y=emfasi(x)
v=[1 -0.97];
y=filter(v,1,x);
```

A.7. winCeps

```
function wc=winCeps(X,Fs,NC)
load condicions;
%N=20e-3*Fs; %mostres d'amplada de la finestra per una finestra de 20 ms.
%separacio=floor(10e-3*Fs); %mostres corresponents a la separacio
%entre finestres de 10 ms.
N=AF*Fs; %mostres d'amplada de la finestra.
separacio=floor(SepF*Fs); %mostres corresponents a la separacio entre finestres.
%nwin=ceil(numel(X)/separacio); %calcul del numero de finestres necessaries.
nwin=floor((numel(X)/separacio)-1); %calcul del numero minim de
%finestres necessaries.
wh=window(@hamming,N); %finestra Hamming d'amplada N mostres.

for i=0:nwin-1
```

```
    for j=1:N

        Xw(j)=wh(j)*X(i*separacio+j); %Xw es una trama de X enfinestrada
%que anirem renovant

    end
    %Calcul dels coeficients cepstrum

    rc=rceps(Xw);
    wc(i+1,:)=rc(1:NC); %copiem els NC primers coeficients del Cepstrum i
%els guardem en una fila
end
```

A.8. deltaCeps

```
function dc=deltaCeps(wc,N)
%construim el vector k dels valors de N
k=(-N:1:N);
%Denominador
denominador=sum(k.^2);
[finestres,nCoefs]=size(wc);
%construim la matriu d'expansio de coeficients Cepstrum + Delta Cepstrum
dc=zeros(finestres-2*N, 2*nCoefs);
%Copia de coeficients Cepstrum (ordre 0)
dc(:, 1:nCoefs)=wc(1+N:finestres-N,:);
ampleD=2*N; %numero de files-1 de la submatriu de wc implicada en
%el calcul dels Delta
for i=1:finestres-ampleD
    subW=wc(i:i+ampleD,:);
    dc(i,nCoefs+1:2*nCoefs)=(k*subW)/denominador;
end
```

A.9. procesaBase

```
%Processa tots els arxius d'audio de la carpeta WAV_Base.
%Els arxius d'audio han de ser de tipus .wav, mono i amb frequencia de
%mostreig de 8000 mostres per segon.
%Un cop processats, els arxius nous es guarden a la carpeta WAV_Proc amb el
%mateix nom.

function procesat = procesaBase(nomDirBase,nomDirProc)
%Fs=8000;
%NC=10; %numero de coeficients cepstrum
```

```
%ND=2; %numero de coeficients a tenir en compte pel calcul dels delta cepstrum.
load condicions;

dirwav=[nomDirBase '/*.wav'];
SP=dir(dirwav);
for i=1:numel(SP)
    nomP=SP(i).name; %nom de la paraula a processar
    nomS=nomP(1:1:end-4); %nom de la paraula sense la terminacio '.wav'
    %nomDir='WAV_Base\'; %nom del directori on es troba la paraula a processar
    nomDirBase=[nomDirBase '\'];
    nomTot=[nomDirBase nomP]; %juntem els noms del directori i la paraula
    nomTproc=[nomDirProc '\' nomS]; %nom de la paraula processada + el directori

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%BLOC PROCESSAT%%%
    %processem paraula

    ptall=wavread(nomTot); %la paraula ja ve tallada de la base de dades
    emf=emfasi(ptall); %filtre pre-emfasi
    w=winCeps(emf,Fs,NC); %enfinestrem y calculem els coeficients cepstrum
    dc=deltaCeps(w,ND); %calculem els delta cepstrum i els afegim als cepstrum
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    save(nomTproc,'dc','ptall'); %guardem els delta cepstrum i la paraula tallada
end
procesat=1;
```

A.10. DTW

```
%DTW(A,B)
% calcula la matriu de costos acumulats del Dynamic Time Warping entre 2
% els vectors A i B
%
function p=DTW(A,B)
%N=numel(A);
%M=numel(B);
N=size(A,1); %numero de files de la matriu A
M=size(B,1); %numero de files de la matriu B
D=zeros(N,M);
%
for i=1:N
    for j=1:M
        %Calculem el cost aplicant la distancia euclidia al quadrat
        %tenint en compte que la distancia euclidea entre 2 vectors x y
        %es D = sqrt(sum((x-y).^2))
```

```
        cost=sum((A(i,:)-B(j,:)).^2);
        %busquem el minim
        if i==1&&j==1
            minim=0;
        elseif i==1
            minim=D(i,j-1);
        elseif j==1
            minim=D(i-1,j);
        else
            minim=D(i,j-1);
            if D(i-1,j)<minim
                minim=D(i-1,j);
            end
            if D(i-1,j-1)<minim
                minim=D(i-1,j-1);
            end
        end
        D(i,j)=minim+cost; %omplim el valor de la matriu D
        %end
    end
end
p=D(N,M);
```

A.11. entraMatriu

```
load condicions;
temps=tempsParaula; %gravarem 3 segons
n=[1:Fs/2]; %mostres de duracio del senyal d'inici 'sINI' (mig segon)
sINI=sin(2*440*n); %senyal sonora inici de cada paraula
%sXIF=sin(1100*n); %senyal sonora numero varies xifres (3aM 5/2 de la freq)
%sCOM=sin(2*660*n); %senyal sonora de coma (5a 3/2 de la freq)
%sNEG=sin(660*n); %senyal sonora de negatiu (5a inferior)
sFIN=sin(2*880*n); %senyal sonora de final de gravacio

%vDim=[1 2]; %vector corresponent al 'size' del vector de files i columnes
%gravem el vector que conte el numero de files i columnes
%wavplay(sINI,Fs);
%grava_Mostra;
%cercaparaula;
%
nomD='Numeros';
%Numero de files de la Matriu
i=1;
nomGB=char(i+48);
```

```
nomDGB=[nomD '\' nomGB];
wavplay(sINI,Fs);
audioGB=grava(nomDGB,temps); %gravem l'arxiu amb el nom al directori
SNR=tallaParaula(nomDGB); %tallem paraula i comprovem si SNR>=0
wavwrite(SNR,Fs,nomDGB); %guardem la paraula tallada amb SNR>=0
i=i+1;
while SNR~=1
    wavplay(sINI,Fs);
    if i<10
        nomGB=char(i+48); %convertim a caracter l'index
    else %hem de separar les dues xifres a caracters
        a=i/10;
        b=floor(a);
        c=(a-b)*10;
        nomGB=[char(b+48) char(c+48)]; %convertim cada caracter i el juntem
    end
    nomDGB=[nomD '\' nomGB];
    audioGB=grava(nomDGB,temps); %gravem el arxiu amb el nom al directori
    SNR=tallaParaula(nomDGB); %tallem paraula i comprovem si SNR>=0
    wavwrite(SNR,Fs,nomDGB); %guardem la paraula tallada amb SNR>=0
    i=i+1;
end
wavplay(sFIN,Fs);
nomW=[nomDGB '.wav'];
delete(nomW);
```

A.12. cercaCalc

```
function vect=cercaCalc(DirecIn)
load condicions;
%DirecIn='Numeros';
DirecNum='WAV_PNum'; %Directori dels numeros
DirecOp='WAV_Poper'; %Directori de les operacions
DirecMod='WAV_Pmod'; %Directori del mode de operacio (Resultat o Numeros)
DIn=dir([DirecIn '/*.wav']);
DNum=dir([DirecNum '/*.mat']);
DOp=dir([DirecOp '/*.mat']);
DMod=dir([DirecMod '/*.mat']);
%
for i=1:1:numel(DIn)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Processem una a una les mostres de numeros
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if i<10
        nomGB=char(i+48); %convertim a caracter l'index
        %nomGB=['0' char(i+48)]; %convertim a caracter l'index
```

```
else %hem de separar les dues xifres a caracters
    a=i/10;
    b=floor(a);
    c=(a-b)*10;
    nomGB=[char(b+48) char(c+48)]; %convertim cada caracter i el juntem
end
nomDGB=[DirecIn '\' nomGB '.wav'];
parlatall=wavread(nomDGB); %llegim la paraula que ja ve tallada
pemf=emfasi(parlatall); %filtre pre-emfasi
pw=winCeps(pemf,Fs,NC); %enfinestrem y calculem els coeficients cepstrum
pdc=deltaCeps(pw,ND); %calculem els delta cepstrum i els afegim als cepstrum
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Busquem dins el directori on tenim paraules processades
%cd WAV_PNum;
if i==1
    directori=DOp;
    DirecRef=DirecOp;
    distancia=zeros(1,numel(DOp));
elseif i==2
    directori=DMod;
    DirecRef=DirecMod;
    distancia=zeros(1,numel(DMod));
else
    directori=DNum;
    DirecRef=DirecNum;
    distancia=zeros(1,numel(DNum));
end

for j=1:1:numel(directori)
    nomB=directori(j).name;
    %paraulaB=wavread(nomB);
    nomBD=[DirecRef '\' nomB];
    load (nomBD);
    %calculem distancia entre paraula mostra i cada paraula de base de dades
    distancia(j)=DTW(dc,pdc); %vector que guarda les distancies
end
%cd .. %tornem al directori amunt
%%%%%%%%%%%%%%%%%%%%
%%Escollim el valor minim de DTW. El minim ens dira quina
%%paraula de la base de dades processades se sembla mes a la paraula que
%%hem rebut. Les paraules processades de la base de dades s'han llegit amb
%%el seu nom d'arxiu en ordre alfabetic. Un cop ordenades alfabeticament,
%%la posicio del maxim dins el vector ens dira cuantes paraules hem de
%%contar fins trobar la paraula buscada.
[minim posicio]=min(distancia);
nomTrobat=directori(posicio).name;
nomTrobat=nomTrobat(1:1:end-4); %eliminem l'extensio del nom
```

```
    if numel(nomTrobat)==1 %passem el nom del numero a format numeric
        numero=double(nomTrobat-48);
    else
        %numero=(double(nomTrobat(1))-48)*10+(double(nomTrobat(2))-48);
        switch nomTrobat
            case 'Negatiu'
                numero=10;
            case 'Coma'
                numero=11;
            case 'Composicio'
                numero=12;
            case 'Suma'
                numero=1;
            case 'Resta'
                numero=2;
            case 'Multiplica'
                numero=3;
            case 'Divideix'
                numero=4;
            case 'Potencia'
                numero=5;
            case 'Numeros'
                numero=1;
            case 'Resultat'
                numero=2;
        end
    end
    vect(i)=numero; %copiem el valor del numero en el vector

end
```

A.13. vector

```
function vect = vector(A)
num=0;
negatiu=10;
coma=11;
compost=12;
decimal=0;
signe=1;
k=0;
i=1;
%files=dim(1);
%columnes=dim(2);
%matC=zeros(columnes,files);
```

```
while i<=numel(A)
    if (A(i)~=negatiu)&&(A(i)~=coma) %tenim un numero
        if A(i)~=compost %el numero sera d'una xifra
            num=A(i);
            num=signe*num;
            k=k+1;
            vect(k)=num;
            signe=1;
        else %sera de varies xifres
            num=0;
            xifres=A(i+1); %numero de xifres
            for j=1:xifres %calculem el numero
                num=num+(A(i+1+j)*10^(xifres-j));
            end
            num=signe*num;
            k=k+1;
            vect(k)=num;
            signe=1;
            i=i+1+xifres;
        end
        %num=signe*num;
        %k=k+1;
        %matC(k)=num;
        %signe=1;
    elseif A(i)==negatiu %sera negatiu
        signe=-1;
    else %sera coma
        decimal=0;
        ndec=A(i+1); %numero de decimals
        for j=1:ndec %calculem els decimals
            decimal=decimal+(A(i+1+j)*10^(ndec-j));
        end %calcul decimals
        decimal=decimal/(10^ndec); %afegim la coma
        if num==0
            if i-2>0
                if A(i-2)==negatiu
                    signe=-1;
                end
            end
        else
            signe=sign(num); %obtenim el signe de la part entera
        end
        num=signe*(abs(num)+decimal); %sumem la part decimal
        %k=k+1;
        vect(k)=num;
        i=i+1+ndec;
        signe=1;
    end %if
```

```
    i=i+1;
end %while
%mat=zeros(files,columnes);
%for col=1:files
%    mat(col,:)=matC(:,col)';
%end
```

A.14. llegirNum

```
function vect=llegirNum(num)
load condicions;
directori='WAV_Num';
base=10; %descomposem en base 10
nDecimals=4; %arrodonim a 4 decimals
negatiu=10;
coma=11;
composicio=12;
%=========================================
%busquem si hi ha coma decimal
decimal=abs(num)-floor(abs(num));
decimal=decimal*10^nDecimals; %eliminem la coma
decimal=round(decimal); %arrodonim a les xifres decimals
ceros=0;
%contem el numero de zeros abans del primer numero decimal
%----------------------------------------
%if (decimal>=1)&&(decimal<10)
%    ceros=3;
%elseif (decimal>=10)&&(decimal<100)
%    ceros=2;
%elseif (decimal>=100)&&(decimal<1000)
%    ceros=1;
%end
%----------------------------------------
for k=0:nDecimals-1
    intervals(k+1)=10^k;
end
k=1;
while (k<=numel(intervals))&&(decimal>=intervals(k))
    k=k+1;
end
if k>1
    ceros=nDecimals-(k-1);
end
%----------------------------------------
if decimal>=10^nDecimals
```

```
    numE=1;
    decimal=0;
else
    numE=0;
end
numD=decimal;
i=1;
if decimal~=0
    %separem el numero en xifres
    quocient=floor(numD/base); %calculem el quocient de la divisio entera
    while quocient>0 %dividim els diferents quocients per la base
        residu=numD-base*quocient; %residu de la divisio entera
        vect2(i)=residu;
        numD=quocient;
        quocient=floor(quocient/base); %actualitzem el nou quocient
        i=i+1;
    end
    vect2(i)=numD;
    %eliminem els zeros que sobren
    j=1;
    while vect2(j)==0
        j=j+1;
    end
    %insertem els zeros que hi havia abans de la primera xifra
%decimal diferent de zero
    if ceros ~=0
        vectCeros=zeros(1,ceros);
        vect2=[vect2 vectCeros];
    end
%    vect=[vect1(end:-1:1) 11 vect2(end:-1:j)];
%else
 %   vect=vect1(end:-1:1);
end %if

%====================================
i=1;
%vect(1)=negatiu;
%i=i+1;
numE=numE+floor(abs(num));

%separem el numero en xifres
quocient=floor(numE/base); %calculem el quocient de la divisio entera
while quocient>0 %dividim els diferents quocients per la base
    residu=numE-base*quocient; %residu de la divisio entera
    vect1(i)=residu;
    numE=quocient;
    quocient=floor(quocient/base); %actualitzem el nou quocient
    i=i+1;
```

```
end
vect1(i)=numE;
%====================================
vectEnter=vect1(end:-1:1);
numXifres=numel(vectEnter);
if (decimal~=0)&&(numXifres>1)
    vectDecimals=vect2(end:-1:j);
    numDecimals=numel(vectDecimals);
    vect=[12 numXifres vectEnter 11 numDecimals vectDecimals];
elseif decimal~=0
    vectDecimals=vect2(end:-1:j);
    numDecimals=numel(vectDecimals);
    vect=[vectEnter 11 numDecimals vectDecimals];
elseif numXifres>1
    vect=[12 numXifres vectEnter];
else
    vect=vectEnter;
end
%si el numero era negatiu especifiquem el signe amb el simbol 10 (negatiu)
if num<0
    vect=[negatiu vect(1:1:end)];
end

%==========================================
%REPRODUIM L'AUDIO ASSOCIAT A CADA NUMERO
%==========================================
i=1;
while i<=numel(vect)
    if vect(i)<10
        nom=char(48+vect(i));
    elseif vect(i)==10
        nom='Negatiu';
    elseif vect(i)==11
        nom='Coma';
    end
    if vect(i)==12 %dictar el numero pero no dictar la paraula 'composicio'
        i=i+1;
    elseif vect(i)==11 %dictar la paraula 'coma' pero no el numero de decimals
        nomTot=[directori '\' nom];
        audioNum=wavread(nomTot);
        wavplay(audioNum,Fs);
        i=i+1;
    else
        %nom=[nom '.wav'];
        nomTot=[directori '\' nom];
        audioNum=wavread(nomTot);
        wavplay(audioNum,Fs);
    end
```

```
    i=i+1;
end %while
```

A.15. calcula

```
function rcalc = calcula(vect)
load resultat;
operacio=vect(1);
opcio=vect(2); %'opcio' decideix si incorpora el darrer resultat o no
if opcio==2
    numeros=[result vect(3:end)];
else
    numeros=vect(3:end);
end
switch operacio
    case 1 %suma
        calc=sum(numeros);
    case 2 %resta
        calc=numeros(1);
        for i=2:numel(numeros)
            calc=calc-numeros(i);
        end
    case 3 %multiplica
        calc=numeros(1);
        for i=2:numel(numeros)
            calc=calc*numeros(i);
        end
    case 4 %divideix
        calc=numeros(1);
        for i=2:numel(numeros)
            calc=calc/numeros(i);
        end
    case 5 %potencia
        calc=numeros(1);
        for i=2:numel(numeros)
            calc=calcˆnumeros(i);
        end
end %switch
vectResult=llegirNum(calc);
result=calc;
save resultat result;
rcalc=[operacio opcio numeros calc];
```

A.16. ensenya_resultat

```
function r = ensenya_resultat(minim,posicio,nomTrobat)
load condicions;
%fprintf('El valor minim es: %d i la posicio es: %d\n',minim, posicio);
%fprintf('La paraula es: %s',nomTrobat);
%cd WAV_Proc;
%load (nomTrobat);
%wavplay(pproc, Fs); %fem que soni la paraula trobada de la base de dades
%r=1;
%cd ..
%%%%%%%%%%%%%%%%%%%%%%%
%fprintf('El valor minim es: %d i la posicio es: %d\n',minim, posicio);
%fprintf('La paraula es: %s',nomTrobat);
%cd WAV_Resultat;
%resultat=wavread(nomTrobat); %llegim l'audio resultat corresponent
%a la paraula trobada
%wavplay(resultat, Fs); %fem que soni la paraula trobada de la base de dades
%r=1;
%cd ..
%%%%%%%%%%%%%%%%%%%%%%%
% Esborrem els numeros anteriors
delete Numeros\*.wav;
%%%%%%%%%%%%%%%%%%%%%%%
nomResultat=['WAV_Resultat\' nomTrobat];
resultat=wavread(nomResultat); %llegim l'audio resultat corresponent a
%la paraula trobada
wavplay(resultat, Fs); %fem que soni la paraula trobada de la base de dades
%%%%
switch nomTrobat
    case char(Funcions(1)) %calculadora
        %clear %Borrar
        %entraMatriu;
        %z=cercaMatriu('Numeros')
        %matriu(z)
        entraMatriu;
        vect=cercaCalc('Numeros');
        vcalcul=vector(vect);
        resCodif=calcula(vcalcul);
        %r=1;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        operacio=resCodif(1);
        switch operacio
            case 1
                oper={'+'};
            case 2
                oper={'-'};
```

```
            case 3
                oper={'*'};
            case 4
                oper={'/'};
            case 5
                oper={'^'};
        end
        memoria=resCodif(2);
        switch memoria
            case 1
                memo={'N'};
            case 2
                memo={'R'};
        end
        for i=3:numel(resCodif)-1 %numeros a calcular
            numch=num2str(resCodif(i)); %convertim a string cada numero
            nums(i-2)=cellstr(numch); %vector de numeros en format cell
        end
        %r=[vcalcul resCodif];
        resCh=num2str(resCodif(end));
        res=cellstr(resCh);
        r=[Funcions(1) oper memo nums res];
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case char(Funcions(2))
        home %Neteja
        r=Funcions(2);
    case char(Funcions(3))
        workspace   %workspace
        r=Funcions(3);
    case char(Funcions(4)) %directori
        dir
        r=Funcions(4);
    case char(Funcions(5)) %preferencies
        preferences
        r=Funcions(5);
    case char(Funcions(6)) %ajuda
        helpwin
        r=Funcions(6);
    otherwise
        r={'No Funcio'};
end
```

A.17. CercaParaula

```
function cerc = CercaParaula()
```

```
%CercaParaula
%Busca l'arxiu d'audio que mes se sembli a un arxiu d'audio anomenat
%'AudioMostra.wav'.
%Compara l'arxiu 'AudioMostra.wav' del directori on es troba
%"CercaParaula.M" amb els arxius d'audio processats de la carpeta
%'WAV_Proc'.
%L'arxiu 'AudioMostra.wav' cal que sigui mono i de frequencia de mostreig
%de 8000 mostres per segon.


%procesaParaules;
%Fs=8000;
%NC=10; %numero de coeficients cepstrum
%ND=2; %numero de coeficients a tenir en compte pel calcul dels delta cepstrum.
load condicions;

DirecRef='WAV_Proc';
PD=dir('WAV_Proc/*.mat');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Processem la mostra de entrada
%parlatall=tallaParaula('AudioMostra.wav'); %tallem paraula
parlatall=wavread('AudioMostra.wav'); %llegim la paraula que ja ve tallada
pemf=emfasi(parlatall); %filtre pre-emfasi
pw=winCeps(pemf,Fs,NC); %enfinestrem y calculem els coeficients cepstrum
pdc=deltaCeps(pw,ND); %calculem els delta cepstrum i els afegim als cepstrum
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Busquem dins el directori on tenim paraules processades
%cd WAV_Proc;
 for j=1:1:numel(PD)
     nomB=PD(j).name;
     %paraulaB=wavread(nomB);
     nomBD=[DirecRef '\' nomB];
     load (nomBD);
    %calculem distancia entre paraula mostra i cada paraula de base de dades
     distancia(j)=DTW(dc,pdc); %vector que guarda les distancies
 end
 %%cd .. %tornem al directori amunt
 %%%%%%%%%%%%%%%%%%%%
 %%Escollim el valor minim de DTW. El minim ens dira quina
 %%paraula de la base de dades processades se sembla mes a la paraula que
 %%hem rebut. Les paraules processades de la base de dades s'han llegit amb
 %%el seu nom d'arxiu en ordre alfabetic. Un cop ordenades alfabeticament,
 %%la posicio del maxim dins el vector ens dira cuantes paraules hem de
 %%contar fins trobar la paraula buscada.
 [minim posicio]=min(distancia);
 nomTrobat=PD(posicio).name;
 nomTrobat=nomTrobat(1:1:end-4); %eliminem l'extensio del nom
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
%MOSTREM EL RESULTAT
%%%%%%%%%%%%%%%%%%%%%%%%
%cd ..
%r=ensenya_resultat(minim,posicio,nomTrobat);
%cerc=[posicio r];
cerc=ensenya_resultat(minim,posicio,nomTrobat);
%fprintf('El valor minim es: %d i la posicio es: %d\n',minim, posicio);
%fprintf('La paraula es: %s',nomTrobat);
%load (nomTrobat);
%wavplay(parlatall, Fs); %fem que soni la paraula trobada de la base de dades
```

A.18. executar

```
function ex = executar()
load condicions;
n=[1:Fs/2]; %mostres de duracio del senyal d'inici 'sINI' (mig segon)
sINI=sin(2*440*n); %senyal sonora inici de gravacio de paraula
wavplay(sINI,Fs);
grava_Mostra;
ex=CercaParaula;
```

A.19. notas

```
function nota = notas(num)
Fs=8000;
durada=0.125;
%n=[1:Fs/2]; %mostres de duracio del senyal d'inici 'sINI' (1/2 segon)
% relacions de notes:
% 2/1 3/2 (5a) 4/3 (4a) 5/4 (3aM o 81/64) 6/5 9/8 (2aM gran) 10/9 (2aM petita)
%16/15 (semito diatonic) 16/9 (7a m)
% En Temperament Igual 1 semito = 2^(1/12). 3 semitons=(2^(1/12))^3=2^(1/4)
% Escala Pitagorica (respecte la fonamental):
% 9/8 81/64 4/3 3/2 27/16 243/128 2/1
%
%Do: 261
%Re: 264
%Mi: 330
%Fa: 349
%Sol: 392
%La: 440
%Si: 494
% Semito en Temperament igual
```

```
r=2^(1/12);

% frequencia de referencia LA
fr=440;

% Canviem la frequencia de referencia a DO
fr=r^-9*fr;
c=r^0*fr*[1/4 1/2 1 2 4]; %Do per les diferents octaves
cs=r^1*fr*[1/4 1/2 1 2 4];
d=r^2*fr*[1/4 1/2 1 2 4];
ds=r^3*fr*[1/4 1/2 1 2 4];
e=r^4*fr*[1/4 1/2 1 2 4];
f=r^5*fr*[1/4 1/2 1 2 4];
fs=r^6*fr*[1/4 1/2 1 2 4];
g=r^7*fr*[1/4 1/2 1 2 4];
gs=r^8*fr*[1/4 1/2 1 2 4];
a=r^9*fr*[1/4 1/2 1 2 4];
as=r^10*fr*[1/4 1/2 1 2 4];
b=r^11*fr*[1/4 1/2 1 2 4];

t=0:1/Fs:durada;

%n1a=sin(2*440*n); %senyal sonora inici de cada paraula
%n2a=sin(1*440*(9/8)*n);
%n3aM=sin(440*n*5/2); %senyal sonora numero varies xifres (3aM 5/2 de la freq)
%n4a=sin(440*n*9/128);
%n5a=sin(2*660*n); %senyal sonora de coma (5a 3/2 de la freq)
%n5a8=sin(660*n); %senyal sonora de negatiu (5a inferior)
%n6a=sin((1/4)*440*n*(3/2)^3);
%n7a=sin(835.135*n);
%n8a=sin(2*880*n); %senyal sonora de final de gravacio
%%n3aM8=sin(2*1100*n);
%n3aM8=sin(440*n*5);
%%n2a8=
%n4a8=sin(440*n*9/64);
%n5a8S=sin(4*660*n);
switch num
    case 0
        nota=g(2);
    case 1
        nota=c(3);
    case 2
        nota=d(3);
    case 3
        nota=e(3);
    case 4
        nota=f(3);
    case 5
```

```
        nota=g(3);
    case 6
        nota=a(3);
    case 7
        nota=b(3);
    case 8
        nota=c(4);
    case 9
        nota=d(4);
    case 10
        nota=e(4);
    case 11
        nota=f(4);
    case 12
        nota=g(4);
end
y=sin(2*pi*nota.*t);
sound(y);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%semito=2^(1/12);
%if num>0
%    nota=sin(((2*440)*semito^num)*n);
%else
%    nota=sin(660*n);
%end
%wavplay(nota,Fs);
```

A.20. ParLab

```
function varargout = ParLab(varargin)
% PARLAB M-file for ParLab.fig
%      PARLAB, by itself, creates a new PARLAB or raises the existing
%      singleton*.
%
%      H = PARLAB returns the handle to a new PARLAB or the handle to
%      the existing singleton*.
%
%      PARLAB('Property','Value',...) creates a new PARLAB using the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to ParLab_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      PARLAB('CALLBACK') and PARLAB('CALLBACK',hObject,...) call the
%      local function named CALLBACK in PARLAB.M with the given input
```

```
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ParLab

% Last Modified by GUIDE v2.5 03-May-2013 02:47:05

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ParLab_OpeningFcn, ...
                   'gui_OutputFcn',  @ParLab_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before ParLab is made visible.
function ParLab_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for ParLab
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ParLab wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = ParLab_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Boto_Configurar.
function Boto_Configurar_Callback(hObject, eventdata, handles)
% hObject    handle to Boto_Configurar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%load condicions;
%nomBase=get(handles.Text_Caixa_Base,'String');
%switch nomBase
%    case 'Funcions'
%        directori='WAV_Base';
%    case 'Numeros'
%        directori='WAV_Num';
%    case 'Operacions'
%        directori='WAV_Oper';
%    case 'Memoria'
%        directori='WAV_Mod';
%    otherwise
%        directori=[];
%end
%if numel(directori)~=0
%    construcBase(directori);
%else
%    missatge=wavread('Error_Config.wav');
%    wavplay(missatge,Fs);
%end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load condicions;
n=[1:Fs/2]; %mostres de duracio del senyal d'inici 'sINI' (mig segon)
sINI=sin(2*440*n); %senyal sonora inici de cada paraula
sFIN=sin(2*880*n); %senyal sonora del final de gravacio de paraules
nomBase=get(handles.Text_Caixa_Base,'String');
switch nomBase
    case 'Funcions'
        directori='WAV_Base';
```

```
        nomProc='WAV_Proc';
        Base=Funcions;
    case 'Numeros'
        directori='WAV_Num';
        nomProc='WAV_PNum';
        Base=Numeros;
    case 'Operacions'
        directori='WAV_Oper';
        nomProc='WAV_Poper';
        Base=Operacions;
    case 'Memoria'
        directori='WAV_Mod';
        nomProc='WAV_Pmod';
        Base=Memoria;
    otherwise
        directori=[];
        Base=[];
end
if numel(directori)~=0
    %construcBase(directori);
    numParaules=numel(Base);
    if numParaules~=0
        missatge1=wavread('ConfigInstruc.wav');
        wavplay(missatge1,Fs);
        for i=1:numParaules %numParaules numero de paraules de la base
            wavplay(sINI,Fs);
            nomParaula=char(Base(i)); %Obtenim el nom de la paraula
%Mostrem el nom de paraula
            set(handles.Text_Caixa_Paraula,'String',nomParaula);
            %set(handles.Text_Caixa_Paraula,'ForegoundColor','red');
            refresh(ParLab);
            grava_Base(directori,nomParaula);
        end %for
        wavplay(sFIN,Fs);
        %fprintf('Iniciant entrenament del sistema\n');
        procesaBase(directori,nomProc);
        missatge2=wavread('Configurat.wav');
        wavplay(missatge2,Fs);
        %fprintf('Sistema Entrenat');
        %construit=1;
    else
        %missatge3=wavread('ErrorBase.wav');
        %wavplay(missatge3,Fs);
        %construit=0;
    end %if
else
    missatge=wavread('Error_Config.wav');
    wavplay(missatge,Fs);
```

```
end


% --- Executes on button press in Boto_Base_Mes.
function Boto_Base_Mes_Callback(hObject, eventdata, handles)
% hObject    handle to Boto_Base_Mes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load condicions;
%Agafem nom de la base de la caixa de text 'Base'
nomBase=get(handles.Text_Caixa_Base,'String');

%Busquem el numero que ocupa la Base dintre del vector Bases
i=1;
while (i<=numel(Bases))&&(strcmp(nomBase,char(Bases(i)))==0)
%strcmp Compare strings.
%strcmp(S1,S2) returns 1 if strings S1 and S2
%are the same and 0 otherwise.
    i=i+1;
end

if i>numel(Bases) %significa que la base no existeix
    num=numel(Bases);
else
    num=i;
end
%Reduim en 1 el numero de la base (ens movem en les
%components del vector Bases)
if (num>=1)&&(num<numel(Bases))
    num=num+1;
else
    num=1;
end

%Etiquetem la base asignant el nom de la base al corresponent numero
%(la component del vector es el nom de la base i la convertim a char)
nomBase=char(Bases(num));

%Mostrem el nom de la base a la caixa de text 'Base'
set(handles.Text_Caixa_Base,'String',nomBase);
%Mostrem el nom de la primera paraula de la base a la caixa de text
%'Paraula'
switch nomBase
    case 'Funcions'
        Base=Funcions;
    case 'Numeros'
        Base=Numeros;
    case 'Operacions'
```

```
        Base=Operacions;
    case 'Memoria'
        Base=Memoria;
    otherwise
        Base=Funcions;
        set(handles.Text_Caixa_Base,'String','Funcions');
end
nomParaula=char(Base(1));
set(handles.Text_Caixa_Paraula,'String',nomParaula);
%%So de la paraula
nota=notas(num);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in Boto_Base_menys.
function Boto_Base_menys_Callback(hObject, eventdata, handles)
% hObject    handle to Boto_Base_menys (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load condicions;
%Agafem nom de la base de la caixa de text 'Base'
nomBase=get(handles.Text_Caixa_Base,'String');

%Busquem el numero que ocupa la Base dintre del vector Bases
i=1;
while (i<=numel(Bases))&&(strcmp(nomBase,char(Bases(i)))==0)
%strcmp Compare strings.
%strcmp(S1,S2) returns 1 if strings S1 and S2
%are the same and 0 otherwise.
    i=i+1;
end

if i>numel(Bases) %significa que la base no existeix
    num=1;
else
    num=i;
end
%Reduim en 1 el numero de la base (ens movem en les
%components del vector Bases)
if (num>1)&&(num<=numel(Bases))
    num=num-1;
else
    num=numel(Bases);
end

%Etiquetem la base asignant el nom de la base al corresponent numero
%(la component del vector es el nom de la base i la convertim a char)
nomBase=char(Bases(num));
```

```
%Mostrem el nom de la base a la caixa de text 'Base'
set(handles.Text_Caixa_Base,'String',nomBase);
%Mostrem el nom de la primera paraula de la base a la caixa de text
%'Paraula'
switch nomBase
    case 'Funcions'
        Base=Funcions;
    case 'Numeros'
        Base=Numeros;
    case 'Operacions'
        Base=Operacions;
    case 'Memoria'
        Base=Memoria;
    otherwise
        Base=Funcions;
        set(handles.Text_Caixa_Base,'String','Funcions');
end
nomParaula=char(Base(1));
set(handles.Text_Caixa_Paraula,'String',nomParaula);
%%So de la paraula
nota=notas(num);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Executes on button press in Boto_Executar.
function Boto_Executar_Callback(hObject, eventdata, handles)
% hObject    handle to Boto_Executar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load condicions;
PD=dir('WAV_Proc/*.mat');
if numel(PD)~=nPS %si es diferent al numero de paraules del sistema
    missatge=wavread('Error_Config.wav');
    wavplay(missatge,Fs);
else
ex=executar;
paraula=char(ex(1));
switch paraula
    case char(Funcions(1))
        operacio=char(ex(2));
        resulCalc=char(ex(end));
        memoria=char(ex(3));
        numeros=[];
        nums=char(ex(4));
        numeros=[nums];
        for i=5:numel(ex)-1 %r=[Funcions(1) oper memo nums res];
            nums=char(ex(i));
```

```
            numeros=[numeros operacio nums];
        end
        set(handles.Text_Caixa_Base,'String',numeros);
        set(handles.Text_Caixa_Paraula,'String',resulCalc);
    case char(Funcions(2))
        set(handles.Text_Caixa_Paraula,'String',paraula);
    case char(Funcions(3))
        set(handles.Text_Caixa_Paraula,'String',paraula);
    case char(Funcions(4))
        set(handles.Text_Caixa_Paraula,'String',paraula);
    case char(Funcions(5))
        set(handles.Text_Caixa_Paraula,'String',paraula);
    case char(Funcions(6))
        set(handles.Text_Caixa_Paraula,'String',paraula);
end

%posic=ex(1);
%PD=dir('WAV_Proc/*.mat');
%paraula=PD(posic).name;
%paraula=paraula(1:1:end-4); %eliminem l'extensio del nom
end

% --- Executes during object creation, after setting all properties.
function Text_Caixa_Paraula_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Text_Caixa_Paraula (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
%%%%%%%%%%
% Pantalla colors originals
%
%if ispc
%    set(hObject,'BackgroundColor','white');
%else
%    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
%end
%%%%%%%%%%
% Pantalla colors Canviats
%
if ispc
    set(hObject,'BackgroundColor','defaultUicontrolBackgroundColor');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function Text_Caixa_Paraula_Callback(hObject, eventdata, handles)
% hObject    handle to Text_Caixa_Paraula (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Text_Caixa_Paraula as text
%        str2double(get(hObject,'String')) returns contents of
% Text_Caixa_Paraula as a double


% --- Executes on button press in Boto_Modificar.
function Boto_Modificar_Callback(hObject, eventdata, handles)
% hObject    handle to Boto_Modificar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load condicions;
nomParaula=get(handles.Text_Caixa_Paraula,'String');
%Agafem nom de la base de la caixa de text 'Base'
nomBase=get(handles.Text_Caixa_Base,'String');
switch nomBase
    case 'Funcions'
        directori='WAV_Base';
        nomProc='WAV_Proc';
        Base=Funcions;
    case 'Numeros'
        directori='WAV_Num';
        nomProc='WAV_PNum';
        Base=Numeros;
    case 'Operacions'
        directori='WAV_Oper';
        nomProc='WAV_Poper';
        Base=Operacions;
    case 'Memoria'
        directori='WAV_Mod';
        nomProc='WAV_Pmod';
        Base=Memoria;
    otherwise
        directori=[];
        Base=[];
end
%comprovem si el nom de Paraula es valid
i=1;
while (i<=numel(Base))&&(strcmp(nomParaula,char(Base(i)))==0)
    i=i+1;
end
if (i<=numel(Base)) %ha trobat una paraula de la base
    n=[1:Fs/2]; %mostres de duracio del senyal d'inici 'sINI' (mig segon)
```

```
    sINI=sin(2*440*n); %senyal sonora inici de gravacio de paraula
    sFIN=sin(2*880*n); %senyal sonora del final de gravacio de paraula
    wavplay(sINI,Fs);
    grava_Base(directori,nomParaula);
    wavplay(sFIN,Fs);
    procesaBase(directori,nomProc);
    missatge=wavread('Modificat.wav');
    wavplay(missatge,Fs);
else
    ms='El nom de la paraula no es correcte';
    errordlg(ms,'Errada en l''entrada de dades','modal')
    missatge=wavread('Error_Dades.wav');
    wavplay(missatge,Fs);
end


% --- Executes on button press in Boto_Paraula_Mes.
function Boto_Paraula_Mes_Callback(hObject, eventdata, handles)
% hObject    handle to Boto_Paraula_Mes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load condicions;
%num=str2double(get(handles.Text_Caixa_Paraula,'String'));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%if (num>=0)&&(num<12)
%    num=num+1;
%else
%    num=0;
%end
%if num<10
%    vnum=char(num+48);
%else
%    num1=floor(num/10);
%   num2=10*(num/10-num1);
%    vnum=[char(num1+48) char(num2+48)];
%end
%set(handles.Text_Caixa_Paraula,'String',vnum);
%nota=notas(num);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nomParaula=get(handles.Text_Caixa_Paraula,'String');
%Agafem nom de la base de la caixa de text 'Base'
nomBase=get(handles.Text_Caixa_Base,'String');
switch nomBase
    case 'Funcions'
        Base=Funcions;
```

```
    case 'Numeros'
        Base=Numeros;
    case 'Operacions'
        Base=Operacions;
    case 'Memoria'
        Base=Memoria;
    otherwise
        Base=Funcions;
        set(handles.Text_Caixa_Base,'String','Funcions');
end
%Busquem el numero que ocupa la paraula dintre de la base
i=1;
while (i<=numel(Base))&&(strcmp(nomParaula,char(Base(i)))==0)
%strcmp Compare strings.
%strcmp(S1,S2) returns 1 if strings S1 and S2
%are the same and 0 otherwise.
    i=i+1;
end

if i>numel(Base) %significa que la paraula no es de la base
    num=numel(Base);
else
    num=i;
end
%Reduim en 1 el numero de la paraula de la base (ens movem en les
%components del vector)
if (num>=1)&&(num<numel(Base))
    num=num+1;
else
    num=1;
end

%Etiquetem la paraula asignant el nom de la paraula al corresponent numero
%(la component del vector es el nom de la paraula i la convertim a char)
nomParaula=char(Base(num));

%Mostrem el nom de la paraula a la caixa de text 'Paraula'
set(handles.Text_Caixa_Paraula,'String',nomParaula);
%%So de la paraula
if strcmp(nomBase,'Numeros')==1
    nota=notas(num-1);
else
    nota=notas(num);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- Executes on button press in Boto_Paraula_menys.
function Boto_Paraula_menys_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to Boto_Paraula_menys (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load condicions;
%num=str2double(get(handles.Text_Caixa_Paraula,'String'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%if (num>0)&&(num<=12)
%    num=num-1;
%else
%    num=12;
%end
%if num<10
%    vnum=char(num+48);
%else
%    num1=floor(num/10);
%    num2=10*(num/10-num1);
%    vnum=[char(num1+48) char(num2+48)];
%end
%set(handles.Text_Caixa_Paraula,'String',vnum);
%nota=notas(num);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nomParaula=get(handles.Text_Caixa_Paraula,'String');
%Agafem nom de la base de la caixa de text 'Base'
nomBase=get(handles.Text_Caixa_Base,'String');
switch nomBase
    case 'Funcions'
        Base=Funcions;
    case 'Numeros'
        Base=Numeros;
    case 'Operacions'
        Base=Operacions;
    case 'Memoria'
        Base=Memoria;
    otherwise
        Base=Funcions;
        set(handles.Text_Caixa_Base,'String','Funcions');
end
%Busquem el numero que ocupa la paraula dintre de la base
i=1;
while (i<=numel(Base))&&(strcmp(nomParaula,char(Base(i)))==0)
%strcmp Compare strings.
%strcmp(S1,S2) returns 1 if strings S1 and S2
%are the same and 0 otherwise.
    i=i+1;
end
```

```
if i>numel(Base) %significa que la paraula no es de la base
    num=1;
else
    num=i;
end
%Reduim en 1 el numero de la paraula de la base (ens movem en les
%components del vector)
if (num>1)&&(num<=numel(Base))
    num=num-1;
else
    num=numel(Base);
end

%Etiquetem la paraula asignant el nom de la paraula al corresponent numero
%(la component del vector es el nom de la paraula i la convertim a char)
nomParaula=char(Base(num));

%Mostrem el nom de la paraula a la caixa de text 'Paraula'
set(handles.Text_Caixa_Paraula,'String',nomParaula);
%%So de la paraula
if strcmp(nomBase,'Numeros')==1
    nota=notas(num-1);
else
    nota=notas(num);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- Executes during object creation, after setting all properties.
function Text_Caixa_Base_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Text_Caixa_Paraula_Base (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
%%%%%%%%%%%%%%%
% Pantalla Colors Originals
%%
%if ispc
%    set(hObject,'BackgroundColor','white');
%else
%    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
%end
%%%%%%%%%%%%%%%
% Pantalla Colors Canviats
%%%
if ispc
    set(hObject,'BackgroundColor','defaultUicontrolBackgroundColor');
```

```
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


function Text_Caixa_Base_Callback(hObject, eventdata, handles)
% hObject    handle to Text_Caixa_Paraula_Base (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Text_Caixa_Paraula_Base as text
%        str2double(get(hObject,'String')) returns contents of
% Text_Caixa_Paraula_Base as a double
```